
FEUP

# P2P Strategies for the Safe Distribution of Rich Media Content

by

Helder Fernandes de Castro

Dissertation submitted to the
Faculty of Engineering of the University of Porto
for the degree of
Doctor of Philosophy
in the
Doctoral Program in Electrical and Computers Engineering

Supervisor
Artur Pimenta Alves (PhD),

Co-Supervisor
Maria Teresa Andrade (PhD)

2012

# Abstract

The production, distribution, and consumption of information goods have faced various challenges over the years. Currently, the status-quo in this field is being shaken by the emergence of the Internet as the prime ground for information exchange.

This powerful new information distribution medium, is very attractive for consumers, who find it much more comfortable. It is also appealing for distributors as it enables a near costless replication and distribution of information commodities. If the on-line medium is combined with P2P distribution, such costs are decreased even further.

However, this near annulment of the mentioned costs, by the Internet, has enabled an explosion of free content exchange. Content scarcity has thus been virtually eliminated in the on-line environment, and is thus challenging age-old intellectual-property regimes

Commercial players have begun adopting all-digital on-line distribution and P2P distribution technologies, inscribed within operations that are characterized by Business Models (BMs) which have simply been transposed from the “real world” onto the on-line environment. These BMs are, typically, very dependent on the scarcity of the delivered content, thus, to make up for its natural loss, in the on-line medium, said BMs generally require the employment of access restricting Digital Rights Management (DRM) technology to artificially recreate the lost scarcity.

In spite of the potential of the Internet medium, attested by the successes of many information content distributing initiatives (legitimate or not), commercial initiatives have generally attained unremarkable results. DRM technologies have frequently been resented and circumvented by users and consequently abandoned by commercial players. P2P distribution technology has also been retreated from, in favour of more traditional ones. Finally, the overall obtained economic results have generally, been unimpressive.

In this work, we set of to discover and identify the root causes of the lack of success of commercial online (specifically, P2P based) content distribution initiatives, and, of the associated, frequent failure of DRM (security) technologies.
We discovered that such root causes are, fundamentally, not technical but mostly economic and are related to the enforcement of inadequate, information scarcity dependent BMs in a scarcity adverse environment. We then identified some BMs which we believe to be adequate for such an environment.
The existing P2P content distribution tools, both in the commercial and non-commercial sectors, are either inadequate or suboptimal, to provide the necessary support for the identified BMs. We thus proceeded to conceive a P2P content distribution architecture, upon which such BMs can be securely implemented.

# Resumo

A produção, distribuição e consumo de bens informacionais têm enfrentado vários desafios ao longo dos tempos. Actualmente, o status-quo neste campo está a ser abalado pelo surgimento da Internet como o terreno privilegiado para a troca de informação.

Este novo e poderoso meio de distribuição de informação, é muito atraente para os consumidores, que o consideram muito mais confortável que anteriores soluções. É atraente também para os distribuidores de conteúdos informacionais, pois permite uma enorme redução dos custos replicação e distribuição desses bens. Se o meio on-line for combinado com a distribuição P2P, esses custos são reduzidos ainda mais.

No entanto, a quase anulação de custos permitida pela Internet, abriu caminho a um crescimento explosivo na troca livre de conteúdos media. A escassez (no sentido económico do termo), de conteúdo informacional, foi assim praticamente eliminada no ambiente on-line, o que coloca sérios desafios aos regimes de propriedade intelectual há muito estabelecidos.

Vários operadores comerciais começaram já a adoptar a distribuição digital e on-line assim como tecnologias de distribuição P2P. Isto tem sido feito no contexto de operações comerciais caracterizadas por Modelos de Negócios (MNs), que foram simplesmente transpostos do "mundo real" para o ambiente on-line. Estes MNs são, normalmente, estritamente dependentes da escassez do conteúdo distribuído. Por essa razão, para compensar a perda natural dessa escassez no meio on-line, os referidos BMs requerem, tipicamente, o emprego de tecnologia de restrição de acesso ao conteúdo, para artificialmente recriar a escassez perdida. A tecnologia em questão é globalmente chamada de tecnologia de Digital Rights Management (DRM).

Apesar do potencial do meio Internet, atestado pelos sucessos de variadas iniciativas (legítimas ou não), na área da distribuição livre de conteúdo informacional, as iniciativas de cariz comercial têm geralmente obtido resultados nada notáveis. As tecnologias de DRM têm-se revelado incómodas para os utilizadores e são frequentemente por eles contornadas. Consequentemente, isto tem resultado no seu abandono pelos operadores comerciais. O emprego de tecnologias P2P de distribuição tem também recuado, em favor de outras mais tradicionais. Por fim, os resultados económicos obtidos têm, em geral, sido inexpressivos.

Neste trabalho, tratamos de investigar e identificar as causas da falta geral de sucesso das iniciativas comerciais de distribuição de conteúdo on-line, (com um enfoque especial naquelas que empregam distribuição P2P), assim como da falha frequente das tecnologias DRM associadas.
Descobrimos que essas causas não são, fundamentalmente, de natureza técnica, mas principalmente económica e que estão relacionadas com o emprego de MNs desadequados por dependerem da preservação da escassez dos bens distribuídos num ambiente adverso a essa escassez. Em face do anterior, procedemos então à identificação de alguns MNs que acreditamos serem adequados para tal ambiente.

As ferramentas, actualmente existentes, para a distribuição P2P de conteúdo, tanto no sector comercial como no não-comercial, são inadequadas ou insuficientes, para fornecer o suporte necessário para os MNs identificados. Para responder a essa necessidade, concebemos uma arquitectura P2P de distribuição de conteúdo, sobre o qual os MNs referidos podem ser implementados com segurança.

# Acknowledgements

This Ph.D. study was undertaken in six years, starting in November 2006, within the context of the Ph.D. program in Electrical Engineering at the Faculdade de Engenharia da Universidade do Porto.
It was supported by the Fundação para a Ciência e a Tecnologia (FCT), during four of those years, and hosted, for the entirety of that period, at INESC TEC under the supervision of professors Artur Pimenta Alves and Maria Teresa Andrade.

I would hereby, like to express my appreciation to FCT, for their important financial support, and to INESC TEC for their welcoming attitude, for providing me with a workplace in a dynamic and fruitful research environment and for their understanding of the specific work conditionments under which a Ph.D. student is.

I would especially like to thank professors Artur Pimenta Alves and Maria Teresa Andrade for their valuable and patient guidance and support throughout the course of these Ph.D. works.

I sincerely thank, also, all the colleagues with whom I have worked and exchanged ideas, during this period, in the context of the elaboration of scientific papers and of my involvement in research projects. Collaborating with them has enabled me to acquire new skills and concepts and to achieve a better divulging of the work that I have developed throughout the Ph.D.

Finally I thank my family, and friends, for their unconditional and ever-present support.

# Contents

**Abstract** ..... **iii**

**Resumo** ..... **v**

**Acknowledgements** ..... **vii**

**Contents** ..... **ix**

**List of Figures and Tables** ..... **xv**

**Symbols and Acronyms** ..... **xvii**

**I Introduction** ..... **1**
1 Context ..... 1
2 Objectives ..... 1
3 Major Results ..... 1
4 Structure of the Dissertation ..... 2

**II P2P Content Delivery Survey** ..... **5**
1 Introduction ..... 5
2 P2P Technology Survey ..... 5
2.1 Introduction ..... 5
2.2 Technical Panorama ..... 8
2.2.1 Introduction ..... 8
2.2.2 Basic P2P Taxonomy ..... 9
2.2.3 Content Discovery and Location ..... 12
2.2.3.1 In Unstructured Architectures ..... 12
2.2.3.2 In Structured Architectures ..... 13
2.2.4 Content Exchange/Retrieval ..... 13
2.2.5 Content Availability ..... 14
2.2.6 Content Management ..... 15
2.2.7 Benevolent Use Enforcement ..... 16
2.2.7.1 Trust Based Incentive Mechanisms ..... 17
2.2.7.2 Trade Based Incentive Mechanisms ..... 19
2.2.7.2.1 Resource Trading Schemes (Barter Based) ..... 19
2.2.7.2.2 Micro-payments Schemes (Bond Based) ..... 20
2.2.8 Security ..... 20
2.2.8.1 Introduction ..... 20
2.2.8.2 Identity Security ..... 21
2.2.8.3 Communication Security ..... 22
2.2.8.3.1 Communication Confidentiality ..... 22
2.2.8.3.2 Communication Integrity and Authenticity ..... 23
2.2.8.3.3 Communication Anonymity ..... 23
2.2.8.4 Content Security ..... 26
2.2.8.4.1 Content Integrity ..... 26
2.2.8.4.2 Content Integrity and Availability ..... 27
2.2.8.4.3 Content Authenticity ..... 27
2.3 Noteworthy Systems ..... 28
2.3.1 Introduction ..... 28
2.3.2 Gnutella ..... 28
2.3.3 BitTorrent ..... 30
2.3.4 eDonkey ..... 32
2.3.5 FastTrack ..... 33
2.4 Summary ..... 35

3 DRM Technology Survey ..... 38
3.1 Introduction ..... 38
3.2 Technological Description ..... 38
3.2.1 Main Logical Mechanisms ..... 38
3.2.1.1 Content Identification ..... 38
3.2.1.2 Content Metadata ..... 39
3.2.1.3 Rights Expression Language ..... 40
3.2.1.4 User and Device Authentication ..... 41
3.2.1.5 Event Reporting ..... 42
3.2.1.6 Content Protection ..... 43
3.2.2 Basic Topology ..... 43
3.3 Specifications and Implementations ..... 47
3.3.1 Specification Initiatives ..... 47
3.3.1.1 OMA DRM ..... 47
3.3.1.2 OpenSDRM ..... 48
3.3.1.3 ISMA/DRM ..... 50
3.3.1.4 MPEG IPMP ..... 51
3.3.1.4.1 MPEG IPMP Extensions ..... 51
3.3.1.4.2 MPEG-21 IPMP ..... 52
3.3.1.5 DMP DRM ..... 52
3.3.2 Implementation Initiatives ..... 53
3.3.2.1 Windows Media DRM ..... 53
3.3.2.1.1 WMDRM Operational Overview ..... 53
3.3.2.1.2 WMDRM Content Protection Scheme ..... 54
3.3.2.1.3 WMRM Licenses ..... 55
3.3.2.2 Helix DRM ..... 55
3.3.2.3 DMDFusion ..... 57
3.3.2.4 Secure Digital Container DRM ..... 58
3.3.2.5 OpenIPMP ..... 60
3.3.2.6 AXMEDIS DRM ..... 62
3.4 Summary ..... 63
4 Commercial P2P Distribution Survey ..... 63
4.1 Introduction ..... 63
4.2 Veoh ..... 63
4.3 Babelgum ..... 63
4.4 JOOST ..... 64
4.5 PPLive ..... 64
4.6 ReelTime ..... 64
4.7 LiveStation ..... 65
4.8 Imeem ..... 65
4.9 BBC iPLayer ..... 65
4.10 Qtrax ..... 66
4.11 Sky Anytime ..... 66
4.12 iMesh ..... 66
4.13 TVUNetworks ..... 66
4.14 Zattoo ..... 67
5 Considerations ..... 67

**III Present Scenario Analysis ..... 71**
1 Introduction ..... 71
2 Overview ..... 71
3 Analysis ..... 72
4 Conclusion ..... 75

**IV Adequate BMs for the New Paradigm ..... 77**
1 Introduction ..... 77

2 Business Models ..... 78
3 Validation ..... 79

**V A Reliable P2P Architecture for the New Paradigm ..... 81**
1 Introduction ..... 81
2 Requirements and Implications ..... 81
2.1 Requirements ..... 81
2.1.1 Introduction ..... 81
2.1.2 Overall Requirements ..... 81
2.1.2.1 Business Requirements ..... 81
2.1.2.2 Rights Protection Requirements ..... 82
2.1.2.3 Usage Requirements ..... 82
2.1.2.3.1 Usage Roles ..... 82
2.1.2.3.2 Content Usage ..... 82
2.1.2.3.3 Monetary Resource Usage ..... 83
2.1.3 Base Requirements ..... 83
2.1.3.1 Operational Requirements ..... 83
2.1.3.2 Security Requirements ..... 83
2.1.3.2.1 Secure Identification ..... 83
2.1.3.2.2 Secure Communication ..... 84
2.1.3.2.3 Secure Content ..... 84
2.2 Architectural Implications ..... 84
2.2.1 Introduction ..... 84
2.2.2 Overall Implications ..... 84
2.2.3 Implications Upon the Core ..... 86
2.2.4 Implications Upon the Peripheral Peers ..... 87
2.2.5 Summary ..... 87
3 Structure ..... 90
3.1 Introduction ..... 90
3.2 Structure Overview ..... 90
3.2.1 Horizontal (or Peer Based) Perspective ..... 90
3.2.2 Vertical (or Layer Based) Perspective ..... 91
3.3 Peer Structure ..... 92
3.4 Peer Structural Roles ..... 93
3.4.1 Introduction ..... 93
3.4.2 Central Core Peer Role ..... 94
3.4.3 Outer Core Peer Role ..... 94
3.4.4 Peripheral Peer Role ..... 95
3.4.5 Summary ..... 96
4 Data Structure and Model ..... 97
4.1 Introduction ..... 97
4.2 Rationale ..... 98
4.3 Registered Information ..... 100
4.4 Data Model ..... 102
4.5 Data Structure Distribution Over the System's Tissue ..... 105
5 Operation ..... 107
5.1 Introduction ..... 107
5.2 Cross Layer Aspects ..... 107
5.2.1 Base Operation Types ..... 107
5.2.2 Operation Identification ..... 111
5.2.3 Peer Identification ..... 114
5.3 IPCL Operation ..... 114
5.3.1 Introduction ..... 114
5.3.2 Interfacing ..... 114
5.4 PLL Operation ..... 114

5.4.1 Introduction ..... 114
5.4.2 Secure Messaging ..... 115
5.4.2.1 Pre-Connection Secure Messaging ..... 115
5.4.2.2 Post-Connection Secure Messaging ..... 118
5.4.3 Peer Registration ..... 122
5.4.3.1 Peripheral Peer Registration ..... 122
5.4.3.2 Outer Core Peer Registration ..... 124
5.4.3.3 Post Registration Actions ..... 124
5.4.4 Peer Registration Update ..... 124
5.4.5 Communication Session Establishment ..... 125
5.4.6 Peer Login ..... 127
5.4.7 PLL Info Discovery ..... 128
5.4.8 Management ..... 130
5.4.8.1 Introduction ..... 130
5.4.8.2 OCP Workload Redistribution ..... 130
5.4.8.3 Infringing or Faulty Peer Behaviour Neutralization ..... 131
5.4.8.4 Peer Shunning ..... 132
5.4.8.4.1 OCP Shunning ..... 132
5.4.8.4.2 PP Shunning ..... 132
5.4.8.5 Peer Readmission ..... 133
5.4.8.6 OCP Updating ..... 133
5.4.8.7 PP Servicing Assignment ..... 134
5.4.9 UEL Info Relaying ..... 134
5.5 UEL Operation ..... 135
5.5.1 Introduction ..... 135
5.5.2 Secure Messaging ..... 135
5.5.2.1 Introduction ..... 135
5.5.2.2 Pre User Login Secure Messaging ..... 136
5.5.2.3 Post User Login Secure Messaging ..... 139
5.5.3 User Registration ..... 139
5.5.4 User Registration Update ..... 141
5.5.5 User Login ..... 141
5.5.6 User Action Monitoring ..... 142
5.5.7 Management ..... 143
5.5.7.1 Introduction ..... 143
5.5.7.2 MO Diffusion ..... 143
5.5.7.3 Infringing or Faulty Peer Behaviour Neutralization ..... 144
5.5.7.4 Infringing User Behaviour Neutralization ..... 145
5.5.7.5 User Shunning ..... 146
5.5.7.6 User Readmission ..... 146
5.5.7.7 OCP Updating ..... 147
5.5.8 User Request Attending ..... 147
5.5.8.1 Introduction ..... 147
5.5.8.2 Simple User Attending Operations ..... 147
5.5.8.2.1 Simple Writing User Attending Operations ..... 147
5.5.8.2.2 Simple Reading User Attending Operations ..... 150
5.5.8.3 Composed User Attending Operations ..... 153
5.5.8.3.1 User Attention Sale Operation ..... 153
5.5.9 DRM Enforcement ..... 154
6 Inter-System Cooperation ..... 155
6.1 Overview ..... 155
6.2 Trust Relationship Establishment ..... 156
6.3 Trust Information Diffusion ..... 157
7 Data and Metadata Objects ..... 158
7.1 Introduction ..... 158

7.2 IPCL Objects ..... 158
7.2.1 IPCL Message ..... 158
7.3 PLL Objects ..... 159
7.3.1 PLL Message ..... 159
7.3.2 Peer Registration Certificate ..... 162
7.3.3 Peer Connection Certificate ..... 163
7.3.4 Peer Info Object ..... 164
7.3.5 Peer Quarantine and Expulsion Lists ..... 165
7.3.6 PLL Info Retrieval Permit ..... 166
7.3.7 PLL Information Location Describing Object ..... 167
7.4 UEL Objects ..... 168
7.4.1 UEL Message ..... 168
7.4.2 User Registration Certificate ..... 169
7.4.3 User Hosting Certificate ..... 170
7.4.4 Search Query Response Objects ..... 171
7.4.5 UEL Information Location Describing Objects ..... 172
7.4.6 UEL Info Retrieval Permit ..... 174
7.4.7 Media Objects ..... 175
7.4.8 MO Ransom Announcement ..... 178
7.4.9 User Monitoring Requests and Responses ..... 178
7.4.10 Inquiry and Inquiry Response Objects ..... 180
8 Exploitation ..... 181
8.1 Introduction ..... 181
8.2 Advertisement BM Support ..... 181
8.3 Donation BM Support ..... 181
8.4 Ransom BM Support ..... 182
8.5 Traditional BM Support ..... 183
8.6 Conclusions ..... 184

**VI Contributions ..... 187**
1 Introduction ..... 187
2 Adequate BM Identification ..... 187
3 Necessary P2P Architecture Definition ..... 189
3.1 Introduction ..... 189
3.2 Comparison to Current P2P Technology ..... 190
3.3 Comparison to Current DRM Capabilities ..... 191
3.4 Comparison to Current Commercial Platforms ..... 192
3.5 Conclusions ..... 193
4 Complex MO Development ..... 193
4.1 Introduction ..... 193
4.2 MO Format Definition ..... 194
4.3 Inter MO Relationships Expression Development ..... 194
5 Publications ..... 195
6 Conclusions ..... 196

**VII Final Remarks ..... 199**

**Annex A – Data Objects ..... 201**
**A.1** IPCL Data Objects ..... 201
**A.1.1** IPCLMHFile ..... 201
**A.2** PLL Data Objects ..... 201
**A.2.1** PLL Message ..... 201
**A.2.1.1** PLLSIHFile ..... 201
**A.2.1.2** PLLPSIHFile ..... 202
**A.2.1.3** PLLPSICTopFile ..... 203
**A.2.2** Peer Registration Certificate ..... 205

**A.2.3** PLL Peer Info Object ..... 207
**A.2.4** Peer Quarantine List ..... 209
**A.2.5** PLL Info Retrieval Permit ..... 210
**A.3** UEL Data Objects ..... 212
**A.3.1** UEL Message ..... 212
**A.3.1.1** UELSIFile ..... 212
**A.3.2** User Registration Certificate ..... 213
**A.3.3** User Hosting Certificate ..... 216
**A.3.4** Search Query Response Object ..... 217
**A.3.5** UEL Information Location Describing Object ..... 220
**A.3.6** UEL Info Retrieval Permit ..... 224
**A.3.7** Media Objects ..... 225
**A.3.8** ERRs and ERs ..... 229
**A.3.9** Inquiry and Inquiry Response IOs ..... 235

**Annex B – P2P Technologies ..... 243**
**B.1** Structured P2P Lookup Protocols ..... 243
**B.1.1** Content Addressable Network ..... 243
**B.1.2** Chord ..... 245
**B.1.3** Tapestry ..... 246
**B.1.4** Kademlia ..... 249

**Annex C – BM Inquiry Results ..... 251**
**C.1** Introduction ..... 251
**C.2** Results ..... 251
**C.2.1** Query Group 1 ..... 251
**C.2.2** Query Group 2 ..... 252
**C.2.3** Query Group 3 ..... 252
**C.2.4** Query Group 4 ..... 253

**Annex D – Data Model Info ..... 257**
**D.1** Registered Information ..... 257
**D.1.1** Introduction ..... 257
**D.1.2** Registered Global Information ..... 257
**D.1.2.1** Registered Global PLL Information ..... 257
**D.1.2.1.1** Registered Global PLL Relational and Procedural Complex Events ..... 257
**D.1.2.1.2** Registered Global PLL Entitary Complex Events ..... 260
**D.1.2.2** Registered Global UEL Information ..... 261
**D.1.2.2.1** Registered Global UEL Relational and Procedural Complex Events ..... 261
**D.1.2.2.1.1** User Originated ..... 261
**D.1.2.2.1.2** System Originated ..... 275
**D.1.2.2.2** Registered Global UEL Entitary Complex Events ..... 277
**D.1.2.3** Registered Global Cross Layer Information ..... 280
**D.1.3** Registered Individual Peer Information ..... 280
**D.1.3.1** Registered Individual PLL Information ..... 280
**D.1.3.1.1** Registered Individual Relational and Procedural Complex Events ..... 280
**D.1.3.2** Registered Individual UEL Information ..... 282
**D.1.3.2.1** Registered Individual Relational and Procedural Complex Events ..... 282

**References ..... 285**

# List of Figures and Tables

Figure 1 – Structure of the Dissertation ..... 3
Figure 2 – Pure P2P Interaction Overview ..... 6
Figure 3 – Decentralized Peer-to-Peer Architecture ..... 10
Figure 4 – Partially Centralized Peer-to-Peer Architecture ..... 10
Figure 5 – Hybrid Decentralized Peer-to-Peer Architecture ..... 11
Figure 6 – Gnutella Structure and Content Search ..... 29
Figure 7 – BitTorrent Protocol Operation ..... 31
Figure 8 – FastTrack Architecture ..... 34
Figure 9 – Content Identification Schemes and Standards (data obtained from [85]) ..... 39
Figure 10 – Model of ER-R Processing and ER Generation (adapted from [102]) ..... 42
Figure 11 – Basic DRM Architecture (adapted from [89]) ..... 45
Figure 12 – Basic Client Structure (adapted from [85]) ..... 45
Figure 13 – Typical DRM Functional Architecture (adapted from [85]) ..... 46
Figure 14 – OpenSDRM Solution Architecture (adapted from [85]) ..... 49
Figure 15 – ISMA DRM Architecture (adapted from [110]) ..... 51
Figure 16 - Windows Media Rights Manager Architecture (adapted from [89]) ..... 54
Figure 17 - RealSystem Media Commerce Suite Architecture (adapted from [89]) ..... 56
Figure 18 – DMDFusion Architecture (adapted from [89]) ..... 57
Figure 19 – SDC DRM Architecture ..... 59
Figure 20 – OpenIPMP Components Diagram (adapted from [85]) ..... 61
Figure 21 – P2PTV overlay network serving several video streams ..... 64
Figure 22 – Typical Commercial P2P Content Delivery System Architecture ..... 69
Figure 23 – P2PTube Structural Overview ..... 91
Figure 24 – P2PTube Horizontal Layers Traversing Peers ..... 91
Figure 25 – Peer Architecture ..... 92
Figure 26 – Example Distribution of OCP Servicing Responsibilities over the PP Collective 95
Figure 27 – Example of Data Structure Section ..... 101
Figure 28 – Basic Data Model ..... 106
Figure 29 – Basic Inter-Peer Interaction Sequences ..... 111
Figure 30 – Cooperation Mesh Constituting an Operation ..... 113
Figure 31 – Pre-Connection Secure Messaging Procedure ..... 115
Figure 32 – Post-Connection Secure Messaging Procedure ..... 119
Figure 33 – Peripheral Peer Registration Process ..... 123
Figure 34 – Inter-Peer Communication Session Establishment Process ..... 125
Figure 35 – Pre User Login Secure Messaging Procedure ..... 136
Figure 36 – User Registration Procedure ..... 140
Figure 37 – User Authentication Procedure ..... 142
Figure 38 – Client/Server User Attending Procedure ..... 148
Figure 39 – Hybrid Operation for MO retrieval ..... 151
Figure 40 – Logical DRM Functionalities in P2PTube Peer Structure ..... 154
Figure 41 – Inter System Trust Spheres Example ..... 156
Figure 42 – Inter-System Trust Establishment ..... 157
Figure 43 – IPCLMHFile Structure ..... 158
Figure 44 – PLLSIHFile Structure ..... 159
Figure 45 – PLLPSIHFile Structure ..... 160
Figure 46 – PLLPSICTopFile Structure ..... 161
Figure 47 – Peer Registration Certificate Structure ..... 162
Figure 48 – Peer Connection Certificate Structure ..... 164
Figure 49 – Peer Info Object Structure ..... 165
Figure 50 – Peer Quarantine List Structure ..... 165
Figure 51 – Peer Info Retrieval Permit Structure ..... 166

Figure 52 – PLL Information Location Describing Object ..... 167
Figure 53 – UELSIFile Structure ..... 168
Figure 54 – User Registration Certificate Structure ..... 169
Figure 55 – User Hosting Certificate Structure ..... 170
Figure 56 – SQRO Structure ..... 171
Figure 57 – MOList Schema Depiction ..... 172
Figure 58 – UELILDO Structure ..... 173
Figure 59 – MO Retrieval Permit Structure ..... 175
Figure 60 – MOTHFile Structure ..... 176
Figure 61 – MOIHFILE Structure ..... 177
Figure 62 – ERR Structure ..... 179
Figure 63 – ER Structure ..... 179
Figure 64 – Contribution Interrelations and Publications ..... 197
Figure 65 – IPCLMHFile Example ..... 201
Figure 66 – PLLSIHFile Example ..... 202
Figure 67 – PLLPSIHFile Example ..... 203
Figure 68 – PLLPSICTopFile Example ..... 205
Figure 70 – Peer Registration Certificate Example ..... 207
Figure 71 – Peer Info Object Example ..... 209
Figure 72 – Peer Quarantine List Example ..... 210
Figure 73 – PLL Info Object Retrieval Permit Example ..... 212
Figure 74 – UELSIFile Example ..... 213
Figure 75 – User Registration Certificate Example ..... 216
Figure 76 – User Hosting Certificate Example ..... 217
Figure 77 – SQRO Example ..... 220
Figure 78 – UELILDO Example ..... 224
Figure 79 – UEL Info Retrieval Permit Example ..... 225
Figure 80 – MOTHFile Example ..... 227
Figure 81 – MOIHFile Example ..... 229
Figure 82 – ERR DID Example ..... 232
Figure 83 – ER DID Example ..... 235
Figure 84 – Inquiry IO Example ..... 238
Figure 85 – Inquiry Response IO Example ..... 241
Figure 86 – CAN Coordinate Space divided between 5 zones (left side image) and 6 zones (right side image) (adapted from [2]) ..... 243
Figure 87 – Chord identifier circle (m=3) (adapted from [2]) ..... 246
Figure 88 – Neighbour Map maintained by a Tapestry Node with ID 67493 (adapted from [2]) ..... 247
Figure 89 – Tapestry/Plaxton Mesh Routing Example using 5 digit long IDs (adapted from [2]) ..... 247

Table 1 – Notation for Figure 22 ..... 69
Table 2 – Requirements to Implications Mapping ..... 87
Table 3 – Peer Responsibilities ..... 96
Table 4 – Notation Definition ..... 107
Table 5 – Publications ..... 195

# Symbols and Acronyms

CDist – A Content Distributor is a commercially operating entity whose activity consists of the distribution of media content.

BM – A Business Model is a conceptual construct which describes the rationale of how an organization creates, delivers, and captures value.

IG – An Information Good is a commodity whose main market value is derived from the information it contains. If it is not bound to a fixed physical body, the Information Good is just the information itself.

IO – An Information Object is the same thing as an information good.

DRM – Digital Rights Management is an activity, developed by hardware or software access control technologies, on behalf of copyright holders and individuals, whose purpose is to protect the rights of such entities, by controlling or limiting the use of their digital property and devices after sale.

PKI – A Public Key Infrastructure is a set of hardware, software, people, policies, and procedures which work together to create, manage, distribute, use, store, and revoke digital certificates.

TTP – A Trusted Third Party is an entity which facilitates interactions between two parties who both trust it.

P2P – Peer-to-Peer is a mode of computer networking, where each computer in the network can directly interact with other network computers either as client or as a server, for the sharing of various resources such as files, peripherals, sensors, etc.

# I Introduction

## 1 Context

The on-going Internet revolution is changing the way that we interact with informational content, and exponentiating the amount of such content at our reach.

This new medium, and associated technologies, has greatly lowered the costs associated to information reproduction, distribution and access, hence, contributing to the democratization of those activities. It has thus been primarily used for the free exchange of content (legitimately and illegitimately). In this field, P2P computer interaction has played a key role, as it optimally exploits Internet's distribution costs reducing capabilities.

The commercial sector is trying to catch up. It has developed various initiatives aimed at exploiting online content distribution. In an initial stage the promoters of these activities have typically tried to do a direct transposition of pre-Internet BMs onto the on-line environment and employed complex and heavy DRM technology to try to enforce "artificial scarcity" onto that environment.

Many such experiments (with different BMs and supporting tools) have occurred and, some of them, are still in operation, employing P2P distribution. However, overall, the field is still immature. Commercial initiatives are lagging behind, free or pirate ones, in terms of their overall popularity, acceptance by the public and overall social impact. Economic failure, altogether, is not uncommon as well.

This problem deserves attention because a purely pirate distribution of content is not sustainable. If those activities grow incessantly they may undermine the production of the very goods that such systems deliver. Furthermore, the tackling of that problem must definitely include, and not refrain from, all-digital on-line delivery (Internet), as this is an extremely powerful medium, whose capabilities for information manipulation and distribution must inevitably be exploited to the fullest.

Thus If we wish to maintain a rich and evolving cultural sphere, while employing the most powerful means at our disposal ever (the Internet), for the distribution of information content, than we need to find solutions to enable a secure and sustainable exploitation of that medium that can assure the rewarding of information content producers.

## 2 Objectives

The purpose of this work is to contribute, with concepts and tools, to the maturing of secure and sustainable on-line content deliver, specifically, over P2P networks.

Our objectives are to thoroughly identify the causes of the disappointing performance of commercial on-line delivering initiatives, (specifically those operating over P2P), and to find adequate solutions to overcome the identified problems.

## 3 Major Results

In line with what is expressed in the previous sections we have identified the main issues that have delayed the progress of secure and sustainable on-line content delivery, to be primordially of an economic nature.

We realized that the entities attempting to attain an economically sustainable online distribution of information content, have frequently failed to securely exploit that medium because they have not correctly assessed the full depth of the changes brought on by the Internet, such as the elimination of information scarcity that it has led to, and have thus been employing inadequate BMs.

We thus performed our own in depth analysis of the present state of things and derived the necessary conclusions. Building on them we defined a set of possible BMs appropriate for a sustainable exploitation of on-line content distribution.

As the tools to optimally support such BMs over a P2P operation mode are missing, we then defined a suitable P2P architecture which offers the necessary tools.

We can thus state that the work conducted during the course of this dissertation has successfully arrived to the formulated objectives. Notably:

- we identified of the main causes preventing a solid adoption of the on-line medium for commercial content distribution (specifically P2P);
- we developed and formalized new BMs, specifically conceived for said on-line operation and, thus, optimally adapted to the new content access paradigm which that medium entails;
- we conceived and designed, to a certain degree of detail, an appropriate technical support structure for those BMs, thus demonstrating the feasibility of sustainable and secure P2P content distribution;
- additionally, we also developed some work on the field media object structuring and enrichment which resulted in an addition to the MPEG-21 international standard.

Accordingly, based on the outcomes of this thesis, we can claim that it is indeed possible to implement successful on-line businesses, which employ P2P content distribution, while taking full advantage of the great potentialities and flexibility offered by the Internet for information content production, distribution and access.

# 4 Structure of the Dissertation

Figure 1 presents a graphical depiction of the various parts of this dissertation, of their role in it and of their interrelations. In Figure 1, white boxes represent a chapter (or a section of it), grey boxes represent annexes, the gradient filled box represents the problem which is identified and addressed in this work, and the rounded corner boxes represent the relationships between the previous “entities”.

Chapter II contains a description of the present state-of-the-art in the field of P2P content distribution, as well as in the associated fields of DRM technologies and P2P distribution employment in commercial initiatives. The contents of this chapter are complemented by those of Annex B.

Chapter III presents our analysis of the above mentioned state-of-the-art. It thus performs the identification of the precise problem/shortcoming, afflicting the present P2P content distribution scenario.

Chapter IV and V present the proposed solutions for the identified problem. The business-oriented component, of such solutions, is presented in chapter IV, whereas the technical component is presented in chapter V. The content of chapter IV is complemented by Annex C, and that of chapter V by Annex A and D.

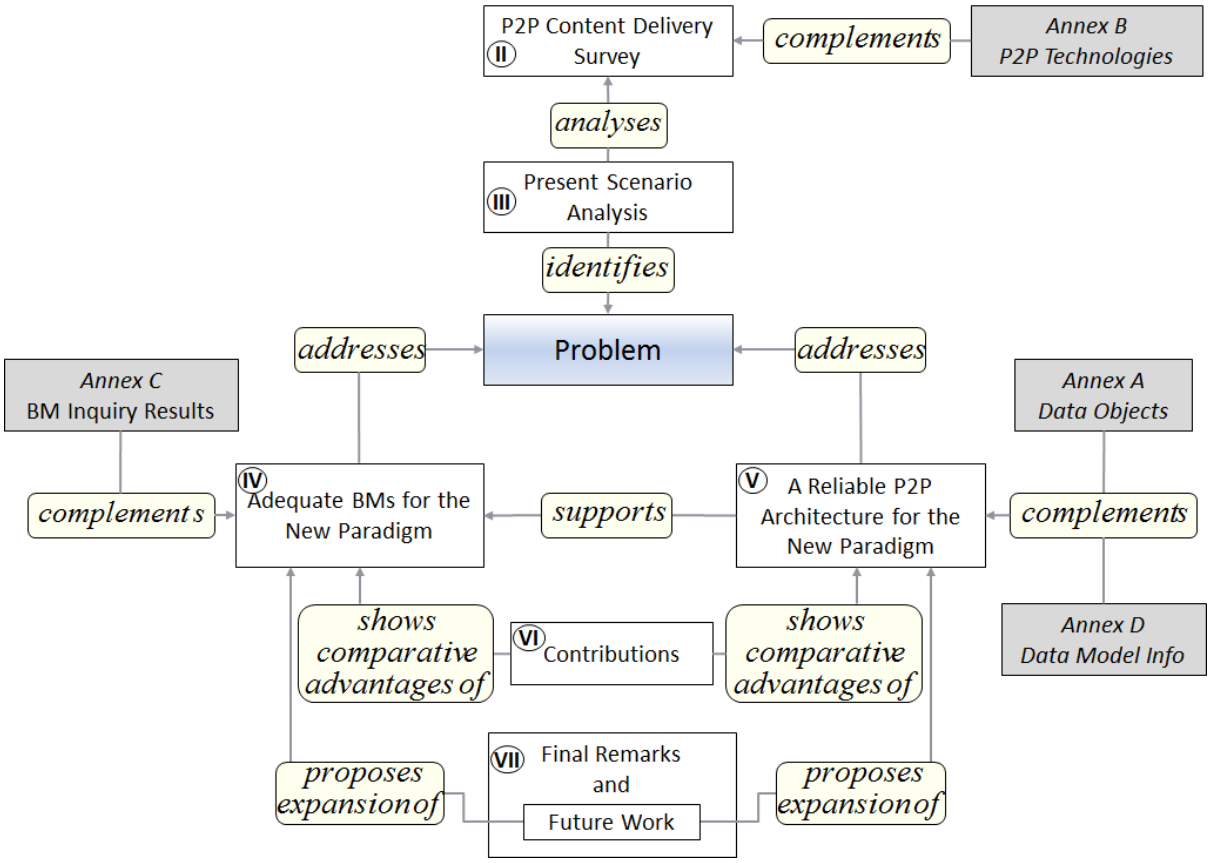


Figure 1 – Structure of the Dissertation

In chapter VI we expose the comparative advantages of our solutions and their specific contributions to the advancement of the state of the art in the field of sustainable and safe distribution of media content over P2P. This chapter also presents the publications that were achieved in the scope of such contributions.

Finally, in chapter VII, we present our concluding remarks and expose the main venues for the future continuation of the developed work.

# II P2P Content Delivery Survey

## 1 Introduction

This PhD work focuses on the development of safe solutions for the P2P distribution of media content. This safeness should be interpreted in a broad manner. It encompasses both reliability of content discovery and retrieval/distribution as well as more strictly security related aspects, such as communicational confidentiality, exchanged data integrity, authenticity and non-repudiability, as well as privacy and anonymity.

The solutions, that are sought for are meant for legitimate P2P content distribution initiatives. These are, typically, commercial ones. Such initiatives involve various aspects, other than just the mere P2P distribution of content, which profoundly impact that distribution. The most relevant of such aspects are the copyright protecting DRM technologies, which are employed in parallel to P2P distribution, and the manner in which both these technologies are combined for the support of specific Business Models. These factors will, thus, guide and condition the present survey.

The survey is divided into three parts. The first one, section 2, describes the state of the art in what regards P2P content distribution. The second, section 3, presents an overview of the current state of development of DRM technology and related initiatives. The third part, section 4, discusses the intersection of the technologies presented in the previous two sections, in the context of current commercial, P2P based, media content distribution initiatives. This last section is not limited to the discussion of technological issues as it includes a short discussion of some economic aspects

## 2 P2P Technology Survey

### 2.1 Introduction

Peer-to-peer (P2P) is a communication model in which each of the intervenient parts has the same capabilities, and either party can initiate a communication session [1], as opposed to the Client-Server model, where each party (server or client) has a different and specific role assigned to it.

A peer-to-peer computer network links its constituting nodes via ad hoc connections and relies mainly on the computing power and bandwidth of the network's participants rather than concentrating it in a reduced number of central nodes (servers). In "pure" peer-to-peer networks, no clients or servers exist and all nodes present equal behaviour, simultaneously operating as both "clients" and "servers" (see Figure 2). In other cases, these systems may have a more hybrid operation, resorting to some centralization for some specific purposes, within the overall P2P operation.

P2P computer networks are typically designed for the sharing of computer resources such as, content, storage and CPU cycles (among others), through direct exchange, rather than requiring the intermediation of a centralized server or entity [2]. The section of these systems devoted to the distribution/sharing of informational content on the Internet, is that which as, recently, received most of the growing, research and public attention surrounding P2P. It is mostly on this subset, of the possible applications of peer-to-peer computer interaction that this survey focuses on.

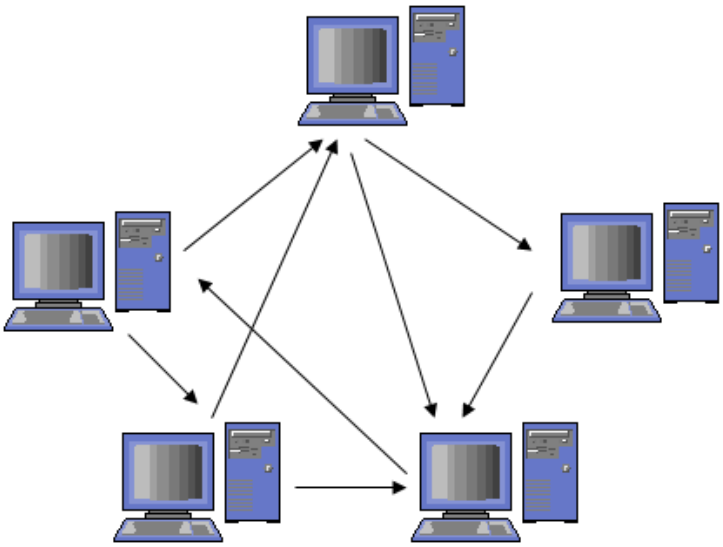

Figure 2 – Pure P2P Interaction Overview

Peer-to-peer is not entirely new. The term P2P in itself is relatively new, but basic P2P technology has existed for as long as USENET and FidoNet since the late 70s early 80s of the 20th century [3].

USENET, was originally created in 1979 by two Duke University graduate students, for the distribution of e-mail-like messages (called "articles") to newsgroups. Files were distributed in batches over phone lines using a flooding algorithm. Consequently, there was no effective way to centralize the functioning of USENET. The resulting application was an extremely decentralized and distributed one [3].

FidoNet, such as USENET, is a decentralized and distributed application for the exchange of messages between users of different Bulletin Board System (BBSs). After experiencing an initial exponential growth and a later shrinking in size because of the closure of many BBSs, it remains in use today [3].

Recently, though, the interest for P2P networking has grown tremendously [4] [5], among Internet users and professionals, especially in the case of file-sharing applications. This renewed and globalized interest began growing in 1999 when an 18-year-old college student developed Napster, which combined the instant-messaging system of IRC, the file-sharing functions of Microsoft Windows and UNIX, and the advanced searching capabilities of various search engines for the facilitation of online digital music files swapping [6]. It became the fastest growing software online, of all time [6].

Such a rapid growth, though, was made possible by three factors which characterized that time [6]:

- growing Internet access bandwidth;
- growing desktop processor power;
- decreasing prices for data storage units.

Napster's facilitation of the transfer of copyrighted material led to its legal prosecution by copyright owners, which filed a law suit against it in December 1999 [7]. Napster was condemned, and after some further attempts for its maintenance online, it was finally terminated in 2002 [8].

Nonetheless, the open space left by Napster's demise was swiftly occupied by numerous other applications/networks, such as FastTrack and Gnutella, and the number of P2P technology users kept rising exponentially [6]. The post-Napster applications adopted a decentralised approach, which made them harder to police (and to target judicially), and have also gained an added range of capabilities.

Distributed lists and searching procedures were developed and are offered by such networks as FastTrack and Gnutella [9]. Content can now be retrieved in separate blocs from different hosts simultaneously, minimizing the strain on their bandwidth, and increasing the availability of the specific content. Even the user's privacy can now be preserved, (up to a certain point), by some systems, such as Freenet, which allows anonymous exchange of content [10].

Promising developments were also made in P2P look-up services based on distributed Dynamic Hash Tables (DHTs), which organize peers into structured overlay networks, assigning each data item to a specific peer, thus allowing much faster searches.

Many promising applications (cooperative file systems, P2P caching, multicast, mobility management, denial-of-service protection, and presence detection), can potentially be built on top of these look-up services.

Furthermore, with the advancement of P2P software and protocols, several of the developed systems, like Gnutella, Gnutella2, EDonkey Network [11], etc., are either interoperable and/or can be accessed through a single interface/application, thus maximizing their overall capacity and attractiveness.

Even though the development of P2P technologies has had to face some legal obstacles, it was not stopped. Relatively recently, P2P technologies have even begun to be embraced by large and legitimate companies, either for the optimization of their inner data structures or to try to tap into its vast potential for the distribution of information rich content such as music or video.

The embrace and development of P2P content distribution in the commercial and non-commercial sectors, has, nonetheless, been uneven. The development of P2P technologies and their employment in non-commercial content distribution initiatives (legitimate or copyright infringing), has proceeded tremendously fast. These solutions are typically developed in an open and cooperative manner, and are freely distributed. This contributes to an accelerated development pace, increased user receptivity and, thus, a reduced deployment time.

In the commercial sector, that process has been much slower. Such initiatives, generally employ either proprietary P2P solutions or pre-existing ones. This sector has not invested a comparable amount of effort on the development of P2P solutions and the efforts that have, in fact been, made, were not done in an environment of openness and interoperability.

The structure of the survey, of the P2P content delivery field, here presented, is composed by the following sub-sections:

- section 2.2 presents the current status of the overall technical field of P2P content delivery;
- section 2.3 presents the most relevant P2P content distribution systems, in existence, which embody the technical provisions described in section 2.2;
- section 2.4 contains a summarizing analysis of the field in scope.

Throughout the presented survey, a special emphasis is placed on the observation of security and reliability aspects of the approached P2P solutions and technologies.

## 2.2 Technical Panorama

### *2.2.1 Introduction*

In P2P computer systems there are two main operational concepts: the peer (or network node or servant), and the communication protocol to which the peer is bound and its associated overall network of nodes.

A peer is a computer application which interacts with other peers through the Internet. A P2P protocol is a set of rules and procedures, which constitutes the language and communication methodology, which the peers of a specific network, use in order to intelligibly interact with each other, thus constituting a P2P network.

Therefore a P2P network is something of a greater ethereality than many other networks. Such a network exists only as a varying set of multiple peers and their common protocol. If peers change the employed protocol, than, the constituted network will be a different one from the original.

P2P computer systems have been employed for a variety of different purposes. The most relevant of those include [2]:

- Communication and Collaboration – Systems providing the infrastructure for the facilitation of direct, (frequently real-time), communication and collaboration between peer computers, such as chat and instant messaging applications;

- Distributed Computation – The purpose of these systems is to take advantage of the distributed processing power available in the network's peers, by breaking down a processing-intensive task into small work units and distributing them by the peers. Such applications of P2P technology inevitably require central coordination, and are thus not purely P2P. Seti@home is an example of such a P2P system;

- Database Systems – These P2P systems work as distributed databases. Several technologies were built within this context such as a scalable distributed query engine which allows relational queries to be performed through thousands of computers (PIER Project [12]), or a metadata infrastructure based in W3C's RDF to provide P2P systems with coherent querying capability (Edutella Project [13]);

- Content Distribution – The purpose of these P2P systems and infrastructures is to facilitate the sharing of digital media and other information rich content between users. Such systems vary from relatively simple direct file sharing applications, to more sophisticated systems which constitute a distributed storage medium capable of enforcing relatively secure and efficient publishing, organizing, indexing, searching, updating, and retrieval of data. Some examples of content distribution P2P systems are: Gnutella, FastTrack, eDonkey, etc.

Presently the most popular application of the P2P computer communication model, is that which is related to the facilitation of media content distribution. Both the public usage and scientific interest for such P2P systems has grown very rapidly in the past years, to the point where the P2P network generated data traffic represents a large portion of the overall Internet data exchange.

Such a great interest as led to an accelerated evolution in P2P technology which greatly advanced and multiplied the number of different available peers, and associated protocols (and thus the number of networks).

The initial ones were the simplest. They merely connected to a central server (or “central peer”) to search for content, and once it was found, it was downloaded from its contributing peer. This was the case, for instance, of Napster, were the sharing of content was performed on a P2P basis, but the content registering and searching process was centralized.

Such solutions were advantageous in terms of content searching efficiency, but presented a single point of failure and “legal attack“, the “central peer” [14].

In a second wave of P2P technology, development a new type of peer and networks emerged. These networks (such as Gnutella), were completely decentralized both in terms of content storing and distribution as well as in terms of content registering and searching. The peers no longer connected to a central peer, but instead established direct communication with other equal peers. When initiating its activity, a peer would connect to a certain number of other peers, which, on their own turn, were connected to other peers etc., in a vast distributed network [14].

In order to search for a specific resource, the inquiring peer would ask all of its neighbouring peers if they possessed it. They would then see if they did, and also pass the message on to all the peers they were connected to, and so forth.

The main advantage of this approach was that it eliminated the central point of failure which existed in previous P2P solutions. The disadvantages included inefficient content searching and the formation of islands of sub-networks that weren't connected to each other [14].

The developments in P2P technology have never stopped, and in a third wave of P2P peers and protocols, many improvements were achieved in terms of increasing the efficiency of content searching and locating, peer “trustworthiness” assessment, removal of “poisoned” or otherwise bad content, increasing download speeds and overall networks reliability, combating free riding, load balancing, etc.

These third wave systems brought some centralization back to P2P architectures. They introduced new concepts such as super-node (or super-peer) or tracker, and have also introduced some differentiation between the peers. Instead of a flat topology of homogenous peers, third wave systems present a somewhat hierarchical structure where some peers take on a more central role and greater responsibilities in the overall coordination of the network.

The regular peers connect to central peers (super-peers, trackers, etc), to obtain from these specific services, such as content location, content retrieval, data forwarding, interaction coordination, etc. The central peers generally interact with each other in a loose and decentralized way, similar to the interaction between the second generation peers.

The next sections (of the enclosing section 2.2), present an overall view of the present P2P technological landscape.

### *2.2.2 Basic P2P Taxonomy*

The existing P2P protocols/networks, and thus, the individual applications that adhere to them, may be subdivided into different categories, according to different characterization vectors. The main such vectors are:

- their level of decentralization – pure peer-to-peer networks would be totally decentralized. Still, in practice these systems have varying degrees of centralization. The three main categories are [2]:

- Purely Decentralized – all of the network's peers operate in exactly the same way, without any centrally coordinating entity. No specialized roles exist for any of the constituting nodes and these are connected in a "flat" layer of peers. Gnutella is an example of such a decentralized P2P system [15], especially before the introduction of ultra-peers (similar to super-nodes). Figure 3 presents an example of a purely decentralized P2P architecture.

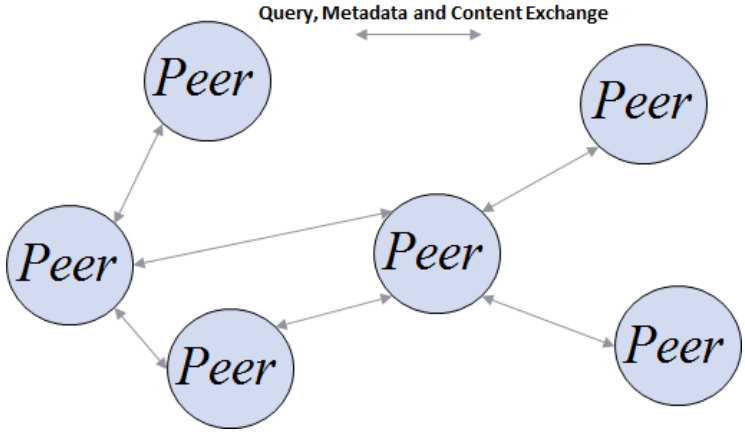


Figure 3 – Decentralized Peer-to-Peer Architecture

- Partially Centralized – some of the network's nodes have more central roles than others, acting as semi-central indexes of the resources which are made available by the local network. These special peers, termed super-nodes, are dynamically selected and the way in which they are organized may vary between networks. A typical example of this type of P2P network is FastTrack [16]. Figure 4 offers a graphical presentation of the architecture of this type of P2P network.

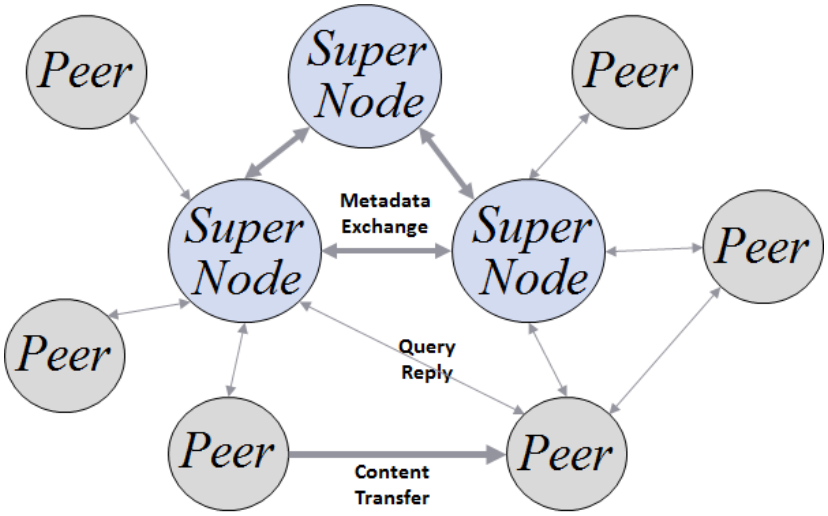


Figure 4 – Partially Centralized Peer-to-Peer Architecture

- Hybrid Decentralized – the interaction between the network's peers is coordinated by a central entity which stores all the resource describing and locating information. After a desired content is found, the source and destination peers contact each other directly for the transfer to take place. The central entity contributes to an optimization of content location and retrieval but also constitutes a central point of failure and bottlenecking which renders this type of P2P systems more vulnerable to censorship or some

forms of malicious attacks. Napster was an example of this type of systems [2]. This type of architecture is depicted in Figure 5.

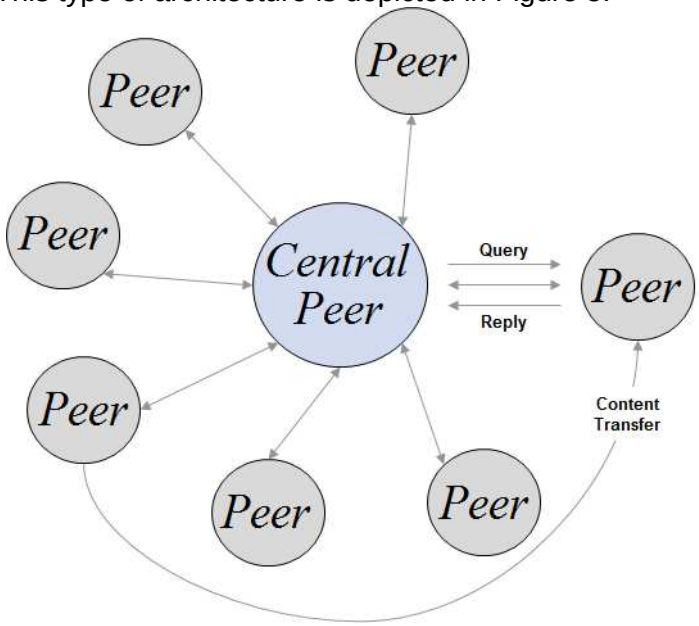


Figure 5 – Hybrid Decentralized Peer-to-Peer Architecture

- their distributed resource placement scheme (or structuring) – P2P overlay networks, may contain a vast number of nodes and a large amount of available resources. The ways in which the nodes are logically connected to each other and how the resources are placed in the network may vary. It may be done in a non-deterministic (ad hoc) way or in a rule adhering way. This difference in the logical topology of the network, and of its resources, is what separates structured from unstructured networks [2]:

    - Unstructured Networks – they are composed of arbitrarily connected nodes and the placement of content, within the network's tissue, is unrelated to the overlay topology. New nodes are easily added just by copying the links of an already connected node and then forming its own links over time. These are resilient to node transience but deal poorly with content discovery and location. Gnutella and FastTrack are two examples of popular unstructured P2P systems [17];

    - Structured Networks – their overlay topology is tightly controlled. Nodes and contents (or pointers to them), are placed at precisely specified locations according to a globally consistent protocol. Such networks perform a mapping between content and location, through the use of distributed routing tables [2]. The main technological approach employed, by P2P networks of this type for its structuring, are the distributed hash table (DHT) based, lookup services. These may be flat or hierarchical [18]. Some notable such protocols are Chord, Pastry, Tapestry, Content Addressable Network (CAN), Kademlia and Tulip. Still, some non-DHT based approaches/networks exist such as Freenet [10], Hypercup [19] and Skipnet [20]. In section B.1 of Annex B some of the above mentioned protocols are explained.
      Structured networks are very efficient at locating content objects whose identifier is precisely known, but have difficulty in enforcing a globally consistent protocol [21].

Structured P2P architectures are typically purely decentralized, with all nodes playing equivalent roles. Unstructured P2P architectures, may present any type of centralism level.

Finally, some proposals exist for hybrid structured-unstructured systems such as the one presented in [22], which is built upon the Gnutella architecture.

### *2.2.3 Content Discovery and Location*

Content discovery is the ability, of a P2P system, to discover about the existence of content objects which satisfy some specific (search) criteria. Content location is the ability of such a system to locate a specific content object in its tissue (peer collective), typically, so that it may be retrieved.

The methods and means through which the two activities are performed are, generally, closely intertwined and, many times, can't actually be separated from each other, in operational terms. Furthermore, the nature and details of such methods typically derives from the type of P2P system where they have their place.

In light of the above, in the following sections we present the fundamental aspects of the state of the art in the fields of P2P content discovery and location (jointly), as a function of the structuredness of the P2P system type where such tools are employed.

#### *2.2.3.1 In Unstructured Architectures*

In unstructured networks the available content needs to be searched-for, given its random location. Typically, in such systems, when a specific content object is discovered (i.e. the existence of a file whose content satisfies some semantic criteria is discovered), it has also been located, as the peer which can match the query to the file, generally, also possesses the file.

There are several methods which can be employed in this type of search, most notably [2]:

- Brute Force Methods – such as flooding the network with propagating queries in a breadth-first or depth-first manner until the desired content is located;
- Sophisticated Methods – such as the use of random walks and routing indexes.

The need for the employment of searching mechanisms for the discovery and location of content in unstructured networks will inevitably affect its global scalability, persistence and availability.

Thus, the main disadvantage, of unstructured P2P systems, within the present context, is that a desired content, even if present, may not always be located, or the process of discovering it may be very inefficient. Recently distributed, or globally highly desired content, is likely to be easily discovered given is wider diffusion through the network. However, rare or old data shared by only a few other peers, is unlikely to be successfully found.

Unstructured systems are thus more appropriate for file-sharing purposes and also for applications where [21]:

- the standard searching operation is keyword based – resources are not precisely identified, and thus the advantages of structured networks are diminished;
- the most desired content is generally replicated at a relatively large fraction of participating peers – content is widely distributed enough to be easily discovered;
- the peer population is highly transient and instable – it is very hard if not impossible to employ a globally consistent protocol to the layout of the network's topology and resource placement and, thus, the use of a structured network becomes impractical.

Nonetheless, even in the cases were the previous characteristics occur, the problem of scalability still resides, especially as the size of the network grows. This is, therefore, one of the subjects still under study in unstructured P2P systems.

Several new search methods and other mechanisms have been proposed to deal with the scalability problems inherent to these systems, such as [2]:

- an algorithm that resorts to the heterogeneity of unstructured systems (most specifically Gnutella) to improve their scalability and search efficiency [21];
- a replacement for the flooding mechanism for searches, which is based on the use of multiple parallel random walks [23];
- the use of proactive replication schemes by copying/distributing content to nodes that did not request it in order to increase the content's diffusion through the system thus facilitating its discovery and retrieval [23].
- a distributed flow control, and topology creation, algorithm which:
    - prevents the overloading of nodes by restricting the flow of queries into them;
    - performs some load balancing (in terms of search query processing), by dynamically evolving the overlay topology to ensure a balanced flow of queries between the nodes.
- the use of more elaborate broadcast policies, which resort to the selection of the neighbouring nodes, to which to forward search queries to, based on their past history, associated to the use of local indices where each peer harbours a list of the data made available by the nodes located within a certain radius from itself [24].

#### *2.2.3.2 In Structured Architectures*

Structured P2P systems are efficiently scalable, (in what regards content discovery and location), in a situation where the exact identifier of the desired content is known. The case of keyword based searching remains somewhat of an issue, for these systems, even though some proposals have been put forward to address it. These are generally based on the use of distributed inverted indexes [25].

The main disadvantage of structured P2P systems is the difficulty of enforcing a globally consistent protocol for node connections, resource placements and efficient query routing in the face of an ever changing node population, where nodes join and abandon the network at any time and not always behave in a benevolent manner [21].

Some of the most relevant lookup protocols, employed by structured P2P systems, for content placement, discovery and location are presented in section B.1 of Annex B.

### *2.2.4 Content Exchange/Retrieval*

P2P systems may use a number of different mechanisms and protocols for the exchange of data (coordination and communication data as well as content data). The HTTP protocol is frequently employed as well as custom develop ones. Several P2P systems and peers also deliver provisions so that peers may be reachable behind firewalls and NAT schemes.

In terms of the way that content files are exchanges between the peers, P2P protocols may employ:

- Bulk content exchange – these systems implement the retrieval of content in one single piece from one single source. After the retrieval is finished the downloaded file's validity may them be validated or not;
- Fragmented content exchange – in such systems, peers retrieve, a specific desired content file, in several fragments from several different other peers. This scheme

generally leads to a greater content availability and allows for a speedier download of content than its bulk exchange counterpart.
Furthermore, fragmented multi-sourced content retrieval allows a more efficient application of bartering schemes, as is the case for the BitTorrent system. In the cases were the P2P protocol enforces a validation of the downloaded content, to combat pollution and data corruption, this content retrieval scheme is also more advantageous as it permits the immediate discarding of corrupted fragments eliminating the need to download the entire content before being able to assess its validity.

### *2.2.5 Content Availability*

The content delivered, by content distribution P2P systems, is distributed and stored through the system's composing peers, which individually are not reliable. In order to ensure the persistence of content access, and to ease its diffusion, these systems resort to the replication of content on a varying number of nodes. This enhances their overall performance in terms of content accessibility and also in what regards resisting censorship attempts and sabotage [2]. Such content replication schemes can be divided into the following types [2]:

- Passive Replication – The content is backed up and thus replicated in every peer which requests it and retrieves it form another peer. This way the content's diffusion occurs in an unenforced way according to natural demand;

- Cache-Based Replication – According to this scheme a specific content is not replicated only in the nodes which requested it but also in all the nodes which, somehow, come into contact with it. For instance, in Freenet [10], upon a successful search and location of a resource, it is transferred back to the requesting node through all the nodes which were initially traversed by the search request. The traversed nodes will all save a local copy of the resource thus increasing its overall availability;

- Active Replication – In this type of scheme, active methods and mechanisms are employed to coordinate, or make more agile, the diffusion of content throughout the network so as to improve the resource's locality and availability;

- Other techniques – Some techniques, such as that used by OceanStore [26] (a project for a global persistent data store), resort to the observation of traffic and content demand to determine the content replication needs, and act upon that in order to satisfy demand.

Data replication creates problems in the fields of data consistency and synchronization (making sure that all the copies of a resource are all equal and that, upon updating of the resource, that all the copies are equally updated), especially in P2P content distribution systems, which are built to permit the deletion and updating of the distributed/stored content. Many times some type of a trade-off must be accepted by maintaining an acceptable ratio between data consistency restrictions and data replication and thus availability [2].

Content replication presents greater problems to structured systems because of the stricter relationship between resource identifiers and locations that takes place in such systems. Nonetheless, solutions have been devised for this, which generally resort to the use of aliases, multiple hash functions, successor-lists or leaf-sets [27].

### *2.2.6 Content Management*

A typical P2P computer system's operation is based upon the sharing of some type of resource between its constituting nodes. Such a resource may be information (typically stored in files), data storage space, data transmission capacity, processing capacity, etc.

As in all systems were resource sharing between multiple independent entities occurs, it becomes necessary to employ some form of coordination, or management, of the access to those resources, and their consumption, so that it can occur in a rational and sustainable way, and so that the system may provide a reliable, efficient and constant service to the nodes which compose it.

The resource management operations more frequently offered are those which permit content location and retrieval. Nonetheless, several other additional operations may be supplied, for instance, for the removal (or usage invalidation) of resources, for the updating of resources, for the creation and maintenance of different versions of a resource, or for the imposition of content consumption restrictions.

It should be noted that the distributed nature of these networks, the unreliability of its composing nodes and the ease with which they can be infiltrated by nefarious peers, makes any task of global coordination difficult.

Within the realm of content distributing P2P networks, the following resource management facilities and practical examples stand out:

- Content Deletion and Update – The enforcing of global content deletion and updating, in a synchronized and coordinated fashion (P2P network wise), is not a simple goal to achieve. Still, some systems offer these, or similar, functionalities. For instance PAST [28], does not truly support delete functionality. Instead, the owner of a file may reclaim the storage space associated to it but it does not guarantee that the file will, globally, cease to be available. Another example is the Publius system. In it the files are published in association with password information, assuring that only the publisher can later delete or change the content. Still this system resorts to a pre-defined set of servers for content storage [29];

- Content Validity Expiration – Some systems provide mechanisms to allow the imposition of validity terms to the information which they distribute. The FreeHaven project enforces document expiration through the use of Rabin's information dispersal algorithm, and the signing of the data shares which carry with them validity time-stamps;

- Content Versioning – This functionality consists of permitting the insertion, and existence, of several different versions of the same content, simultaneously, in the system, in a controlled, organized and reliable way. OceanStore [26], for instance, offers such functionality by employing universal unique data object identifiers, data encoding and redundant and ring-structured distribution and document id lists maintenance;

- Directory Structure – Some systems provide an abstraction layer, above content access, which works as a global directory structure. This way, they provide a mask, which hides the distributed nature of the network and of content location. Mnemosyne [30] is one such system.
  The WayFinder system also supports a distributed directory structure over a P2P infrastructure [31], by overlaying the individual peer's directory structures on top of one another resorting to name or semantic proximity. Ivy [32], implements a multi-

user read/write P2P file system. It employs a set of logs, (one log per participant), stored in a DHash distributed hash table. Data is located and retrieved by consulting all logs, yet data modification is achieved through the modification of only the log of the modifying peer;

- Content Searching – Content discovery capabilities may also exist, in varying performance and efficiently levels. Unstructured P2P topologies (Gnutella, Kazaa), provide keyword based search functionalities whose use is intuitive but not efficiently scalable. Structured systems provide very efficient content discovery facilities, but these are based on rigid file identifiers.
  The development of mechanisms for the provision of keyword based search facilities operation, upon an exact-match query sub-layer is something that is still to be satisfactorily handled;

- Data Storage and Bandwidth Management – This consists of the managing of data storage space and bandwidth usage by the peers. The amount of disk space that is to be contributed with, by each peer, is generally something that each one of them can specify independently. MojoNation allows users to donate data storage capacity in exchange for economic or other type or rewarding. The PAST system [28] employs a secure quota system. The system's users are either attributed a fixed quota of data storage capacity, which they can consume, or alternatively they can take advantage of as much capacity as that which they contribute themselves, on their node. The PAST system also proposes an optional extension of itself, for the incorporation of reliable organizations, named brokers. These are able of trading storage capacity and of issuing smartcards to users, which control the storage capacity usage and supply, that a user can or must perform respectively.
  Besides the quota system, these schemes may employ additional methods to prevent or dissuade nefarious events, such as denial of service attacks. Publius, for instance, employs such methods. Its publishers are requested to perform computational work, solving a mathematical problem, before obtaining permission to publish content.
  The Kosha system, is a proposal for a P2P enhancement of network file systems. According to its operating scheme, the participating nodes are organized into a structured P2P overlay. It employs NFS facilities to make files available throughout the network assuring file location transparency [33];

### *2.2.7 Benevolent Use Enforcement*

P2P systems are especially dependent on the voluntary and benign participation of their users and constituent nodes, in order to provide a reliable and efficient service. This makes it necessary to employ mechanisms which promote and stimulate a benign and cooperative behaviour, from the users and peers, as well as to provide some support for the accountability of actions performed and for the "punishing" of sabotaging or selfish behaviour [2].

Uncooperative behaviour, in P2P systems, generally consists of the so-called "free-rider" attitude, where users/nodes only consume the system's resources and do not contribute back with any. Sabotaging behaviour, on the other hand, comprises a vas set of possible actions, like:

- Poisoning and polluting attacks, where corrupted, or somehow altered content, which does not correspond to its description, is injected into the system thus rendering the search for desired assets more inefficient;
- Insertion of viruses or malware into carried data;

- Denial-of-service attacks where some portion of the network (or possibly all of it), is deliberately overwhelmed with inflowing information by other network peers which take advantage of their inside knowledge of the system;
- Spamming, where unsolicited information is sent across the network;
- Collusion attacks, where multiple nodes cooperate in order to, somehow, harm, elude or manipulate the network for their own selfish or sabotaging purposes.

Such behaviour may be motivated by any number of reasons, such as:

- the desire to manipulate the systems for some kind of inappropriate, individual gain;
- vandalism;
- the attaining of some gain, coming from the damaging of the network's operability.

P2P solutions lacking the referred mechanisms will most likely experience systemic problems ranging from significant degradation of performance to unreliable and untrustworthy availability of resources, or even to complete system inoperability.

Still, it is technically very difficult to promote incentives for fair use and to provide accountability in peer-to-peer systems. The transient population of users/peers, the difficulty to perform peer identification and to maintain trustworthy records about peer's past behaviour and the level of decentralization of these systems makes it so [2].

Two main types of solutions have been put forward for the enforcing of fair and benevolent use in P2P systems [2]:

- Trust-based Incentive Mechanisms – These mechanisms use trust as the determining factor for the establishment of a P2P transaction. Users/peers will be prompted to maintain a posture which will earn them the confidence of the other peers and thus contribute to better servicing of their requests. The principal example of such mechanisms are the Reputation Management mechanisms;
- Trade-based Incentive Mechanisms – When these mechanisms are employed, whenever a peer is sharing some resource, and thus servicing the system, it is immediately rewarded by the system, either directly or indirectly, and vice versa. This category is fundamentally composed by various micro-payment mechanisms and resource trading schemes such as Bartering Mechanisms.

The next sub-sections focus on the main types of Trust-based Incentive Mechanisms (Reputation Mechanisms) and Trade-based Incentive Mechanisms.

#### *2.2.7.1 Trust Based Incentive Mechanisms*

In trust based incentive mechanisms, a peer accedes to cooperate, or fulfil another peer's request, if it trusts the requesting peer (or if its trusted peers trust it). In this way, the serviced peer is not compelled to explicitly reward the servicing peer. The service is performed because in that way the servicing node earns the trust of the serviced peer and of the community/network. That earned trust will later be beneficial to it when it seeks to be serviced by the community.

Enabling of this type of incentive system requires the systematic evaluation of the behavioural history of the peers (or their reputation), by each other, and thus implies the existence of some reputation computation infrastructure or mechanism. These are called reputation management systems.

Such systems operate as organized, inter-peer, information exchange networks, where the transacted data describes the perception, by the emitting peers, of the behaviour of the other such nodes, with which they interact. This is done in a way so as to permit that each node

may obtain a notion of the reliability and trustworthiness of the surrounding mean (P2P system).

These mechanisms can be implemented in a relatively easy and reliable way, when a trusted centralized entity exists, which can be resorted to for the coordination, monitoring and information delivering, concerning node behaviour. Still, this is not the case for the typical P2P systems.

In most peer-to-peer networks, given their distributed nature and the difficulty in unequivocally identifying peers, no single, recognizable entity exits which is in a position to manage and distribute reputation information. Therefore such information must also be managed in a distribute way throughout the network.

The main purpose of P2P reputation management systems is to perform or facilitate the aggregation of reputation information which is obtained by the individual peers, in the regular interaction with each other, and to it make available, throughout the entire network, in a safe and scalable way, thus permitting the formation of a global reputation rating for the system's constituting peers.

Several such mechanisms have been proposed and/or implemented [2]. A few examples are presented below:

- The EigenTrust algorithm, which compiles global reputation data about peers based upon their history of uploads. This data is distributed through the network's peers which they use in order to evaluate each other's credibility before engaging in exchanges. Such global reputation values are calculated using the local reputation values which peers assign each other and weighing them by the global reputation of the assigning peers [34];

- A partially centralized mechanism has been put forward in [35]. This system uses central reputation computation agents and data encryption, in order to securely manage and locally store, reputation information. Two different ways were conceived for the calculation of reputation values, a credit/debit and a credit only scheme;

- PeerTrust is a decentralized reputation mechanism presented in [36]. According to this scheme, a trust value for a peer is calculated based on: the level of satisfaction proportioned to other peers, by that peer; the total number of transactions in which the peer was involved; and a balancing factor to attenuate the effects of malicious reports. The reputation data is stored in the network in a distributed fashion;

- Approaches based on the peer's utility to the system, instead of on its cooperativeness and fairplay, have also been proposed. These mechanisms resort to the calculation of a peer's utility, based on the amount and popularity of content stored and shared by the peer [37];

- Distributed auditing mechanisms have also been considered for the enforcement of fair and benevolent use. In such schemes, nodes will more or less randomly audit each other's logs, comparing them with those of every node with which the audited node claims to have share a resource. This way, prevaricating nodes will be detected. This scheme implies some way to unambiguously and universally identify peers;

- A proposal also exists for combined use of the reputations of peers and resources [38], which provides a finer grained picture of the credible assets in a P2P system.

Just like other components of P2P architectures, reputation systems are also the target of attacks. The purpose of such attacks is not to directly affect the availability of resources, but mainly to hamper the system's ability to provide accountability for malicious behaviour on the part of the participating peers. These attacks may be of multiple types. Some of the most relevant ones include:

- a peer joins the system and behaves in a correct manner long enough to obtain a good reputation. It them starts behaving inappropriately;
- groups of peers collude in order to provide each other undeserved positive reputations;
- isolated peers, or groups of them, cooperate to damage the reputation of other peer(s), for instance through "ID stealth" behaviour [38];

- the peer(s) exploit the use of pseudonyms, by P2P systems (given the lack of a central peer identifying entity), by creating and controlling multiple identities, in order to be able to discard the bad reputation earned into some of the identities, while leaving the reputation of the others untouched.

#### *2.2.7.2 Trade Based Incentive Mechanisms*

In trade based incentive mechanisms, a peer accedes to cooperate, or fulfil, another peer's request if it is explicitly rewarded for that. Such reward consists of a resource/service that the requesting peer supplies/performs for the cooperating/servicing peer, and in this way, both the peers obtain a gain from the transaction.

While in trust based systems, confidence is earned throughout time and can be later profited from, whenever desired. In trade based incentive mechanisms, when a peer wishes to consume some resource from the system it must contribute with a resource of its own, more or less, immediately.

Trade based incentive mechanisms can be classified as either providing immediate reward (barter trade based), or deferred reward (bond based).

Generally the barter trade based mechanisms involve the immediate mutual exchange of resources (files), and thus, can be called resource trading schemes. The bond based schemes involve the exchange, between peers, of "*bonds*" in which the requesting peer promises an action in return to the servicing peer [31]. These can be considered micro-payments schemes.

##### *2.2.7.2.1 Resource Trading Schemes (Barter Based)*

This type of trading schemes, which, as previously explained, imply a direct mutual exchange of resources between the peers, are arguably the most appropriate for insuring the sustainability of peer-to-peer economies and communities/systems in their initial stages of existence [39]. Several such schemes have been proposed or implemented.

One of the most notable, of such mechanisms, is that which is implemented in BitTorrent [40]. In the BitTorrent system the resources (files) are broken into several smaller pieces. Each peer, who desires to obtain the entire resource, must retrieve all the pieces from other peers, which already have them. In the process, the peers reciprocate by uploading the fractions of the resource, which they have, to peers which don't have it, and which are uploading to them. In this way, the peers form bidirectional content delivering connections implementing the bartering system.

In [41] a bartering system is presented, which proposes an extension to simple one-to-one exchanges, in order to permit transitive chains of peer exchanges, where n-way transactions take place between rings of peers. It also proposes an algorithm for the detection of the content exchange rings

Another resource trading mechanism is proposed in [42], where peers acquire resources, from other peers in the network, by trading with their own resources. The main resource, in which the work is focused, is storage space/data. However, it also theorizes the extension of the tradable goods to processing cycles or even to abstract resources, as well as the trade of one type of resource for another.

#### *2.2.7.2.2 Micro-payments Schemes (Bond Based)*

Several centralized and decentralized micro-payment (or bond), based incentive systems have been developed or proposed for P2P networks. Some possible examples are:

- A decentralized bond based incentive mechanism is that which is employed by FreeHaven. The nodes which constitute this network establish contracts, to store each other's data, for a certain period of time. By honouring the established contracts, a node increases its reputation, and thus the chances of the other peers honouring its commitments to it [43];

- In [44] a bond based incentive system is presented, whose operation is based on the maintenance of a distributed list of the resource consumption and contribution of each participating peer. The accumulated credits of a node increase as it contributes with resources, and decrease as the node consumes resources;

- A proposal for a peer-to-peer system where each peer is a software agent and where all the agents cooperate for mutual benefit is laid forth in [45]. Two pricing mechanisms are presented to motivate each agent to maintain a rational and self-advantageous behaviour while at the same time safeguarding systemic efficiency;

- In [46] a micropayment system, PPay, is presented, which takes advantage of the specific characteristics of P2P networks, to optimize its operation and assure security, guaranteeing that all fraud is detectable, traceable and unprofitable through the employment of the concepts of floating and self-managed currency.

Micro-payment schemes fall under two main categories [47]:
- Fungible – in these payment types, peers are paid with something redeemable, that may later be used in exchange with other peers;
- Non-fungible - in these payment types, peers are paid with something that cannot be traded with other peers.

### *2.2.8 Security*

#### *2.2.8.1 Introduction*

P2P content distribution systems are voluntarily formed and have a distributed, open and autonomous nature. These networks are constituted by inherently selfish and unreliable nodes (or even non-benevolent nodes), whose behaviour is erratic and unpredictable. In such conditions it is hard, for such systems, to provide a secure peer interaction and content exchange environment. Some mechanisms must, thus, be put in place, so that P2P networks may achieve some operational security and reliability.

The main concerns generally addressed by such mechanisms are [2]:

- Identity security – these mechanisms are meant to enable the maintenance of identity verifiability for peers and users;
- Communication security – these mechanisms are meant to enable the maintenance of communicational confidentiality, integrity, authenticity and anonymity;
- Content security – these mechanisms are meant to enable the maintenance of content integrity, authenticity, availability, and access control;

The security related mechanisms, of P2P systems, frequently involve cryptographic means [2]. These are based on either symmetric/secret key or asymmetric/public key schemes, or combinations of the two:

- secret key schemes operate based on a common knowledge of a secret key by the sender and receiver/storer. The key is used for encryption and decryption of data and message authentication. These mechanisms require that the keys are separately and a priori distributed in an out of bound procedure [6];
- public key based mechanisms use asymmetric key pairs. One of the keys is publicly distributed, while the other is kept private. In these schemes no need for an out-of-band key distribution structure exists, nonetheless it is still necessary to assure the safe distribution of public keys [6].

#### *2.2.8.2 Identity Security*

Distributed systems, such as P2P networks, which have a very transient population, allow for a single physical entity, (peer or user), to participate in it, under several different identities. This fact poses a set of security challenges, related to access control, authentication (or verifiable identification), and reliability of identity management.

These security issues may have an especially negative impact on P2P systems which employ content replication or fragmentation schemes, for the distribution of content, across multiple nodes, for security and availability.

The use of multiple identities by a single physical entity for nefarious purposes or to engage in selfish behaviour is known as a “Sybil Attack". Given its relevance, this problem has received a fair deal of attention, and some solutions have been proposed.

Some solutions are based on the consumption of identification challenges, for the deterrence of identity forging. These however are considered unrealistic and impractical [48], seen as they presuppose that:

- all peers operate under nearly identical resource constraints;
- all identities presented by peers are validated simultaneously by all the participating peers;
- when accepting identities that are not directly validated, the required number of vouching peers exceeds the number of system wide failure.

Other proposals exist, for enabling P2P systems to support peer registration (and subsequently, some anonymity provision, content authenticity verifiability and content availability), which are generally based on distributed algorithms, more or less independent from any fixed central entity. The most robust of such proposals employ some form of collective/mutual authentication between peers. Some of these are:

- threshold cryptography [49], (which is not purely distributed as it requires a trusted third party);
- PGP [50] (which requires human evaluation of trustworthiness);
- “trusted peer group” based authentication [51].

Further solutions, yet, are proposed:

- in [52] a proposal is made for a self-registration based mechanism, where nodes calculate their identifier based on their IP address and port, and register it in the P2P network itself, at already successfully registered nodes;
- in [53] a method is proposed to deal with flooding and DoS attacks, and thus deter inappropriate access, which is based in the issuing of a resource consuming cryptographic puzzle to the requesting peer, when the contacted peer suspects that it is under attack, so that the attacker spends more resources than the victim. This technique can be used to defend against “Sybil Attacks” by overburdening an attacker;
- in [54] a system is presented which provides access control enforcing such capabilities as reading and writing restricting to only authorized users, through the employment of data encryption and the provision of updated keys to authorized recipients.

Systems or proposals such as those presented in [55] and [56] employ one form or another of a central authority for the assignment of identities.

In private P2P networks, given their smaller dimensions and the fact that the involved users typically know each other, it is possible for users to safely exchange cryptographic keys and identifiers out-of-band. These parameters may then be employed to perform the mutual authentication of peers and users.

Most of the above presented solutions (those which do not employ any trusted central authority for identity provision), however robust, still present considerable weaknesses in the face of sufficiently vast and sophisticated attacks, and (in the case of the collective/mutual authentication based ones) lead to a considerable operational overhead. In truth, it can be conclusively stated that an implicit, or explicit central certification or identification authority is inevitably necessary to assure that P2P systems are impervious to a “Sybil Attack", and thus provide access control, authentication and identity management [48].

#### *2.2.8.3 Communication Security*

Communicational security provisions mean to assure the confidentiality, integrity, authenticity and anonymity of information packages exchanged between peers.

##### *2.2.8.3.1 Communication Confidentiality*

The confidentiality of inter-peer communications naturally implies some form of encoding of the exchanged messages, and this means that the communicating parties need to share some form of secret, so that the messages may be encoded and decoded.

Different schemes exist to provide P2P systems with varying levels communicational confidentiality. Some of the most illustrative examples, of the current technical panorama in this area, are the following:

- in [57] a “Trusted Computing” based solution is proposed, which employs tamper resistant security hardware. It presupposes the assistance of centralized provisions, be it the hardware provider or some other centralized trust providing entity;
- in [58] a solution is proposed which employs a quorum based decentralized PKI, which is integrated in the P2P structure;
- in private P2P networks, as users know one another, it is feasible for them to securely exchange cryptographic keys out-of-band. These keys may then be employed to cipher the exchanged messages.

#### *2.2.8.3.2 Communication Integrity and Authenticity*

The provision of communication integrity and authenticity, in P2P systems, is a similar question to assurance of content integrity (see section 2.2.8.4.1). As such the developed solutions are very much the same.

Furthermore, such provision also involves secure routing techniques, and protocols. The purpose of the latter is to prevent malicious peers from corrupting, deleting, or denying access to the data stored by, or exchanged through P2P systems.

According to [59], for the provision of secure routing, in a P2P system, the fundamental issues that must be addressed are:

- Secure assignment of IDs to nodes – which can be robustly achieved through the use of a trusted central authority. It may also be performed through distributed algorithms, which employ collective/mutual authentication between peers (see section 2.2.8.2). This alternative, however is less robust;
- Secure maintenance of routing tables – this can be achieved, with relative robustness, by imposing strong constraints, and demands, on the set of nodes which can be allowed into the routing table [59];
- Secure forwarding of messages – this can be achieved, with relative robustness by verifying the proper delivery of messages and resorting to the use of diverse and redundant routing.

#### *2.2.8.3.3 Communication Anonymity*

Public interest in P2P anonymity has grown rapidly in the recent past. The main concern of P2P users/peers who wish to remain anonymous is not to be identifiable as a sender or receiver, of information.
There are a variety of reasons for this, such as:

- Preservation of privacy – The user does not wish for his activity to be tracked or for it to be known that he/she possesses or is looking for a specific informational content;
- Censorship circumvention – The user wishes to evade local, organizational, or governmental censorship to the distribution or access to some information;
- Avoiding social condemnation – The content that the user is distributing or searching for, is legal but it is socially condemned, embarrassing or unaccepted in the user's social mean, workplace, religion, etc.;
- Fear of reprisals – The user may which to distribute or consume content, which due to its nature, (activist, denunciatory, etc.), may lead to reprisals on the user if his identity is known.

Anonymity, in P2P computing, is the network's capability to hide the connection between an observable action (for instance, a query message traversing the network), and the identity of the entities involved in this action [60].

In such content distribution systems, anonymity can be of different types [60]:

- Sender anonymity to any node or a global attacker;
- Responder anonymity to any node or a global attacker;
- Sender-responder anonymity to any node or a global attacker.

Anonymity can also involve the identity of content storing nodes, details about content's nature, and the specifics of queries.

Furthermore the level of anonymity provided by a system can also vary between the following categories [60] [61]:

- Beyond suspicion (BS) – From the perspective of an observing node, the detected peer presents no greater probability of having being the source of an action than any other node;
- Probable innocence (ProbI) – From the perspective of an observing node, the detected peer presents no greater probability of having being the source of an action than of not having being;
- Possible innocence (PossI) – From the perspective of an observing node, there is a non-negligible probability that the detected peer did not originate a specific action.

In anonymous P2P networks, the peers are identified by pseudonymous by default, and the fundamental difference between them and non-anonymous networks, lies in the way in which information is routed and transmitted through them.

In such anonymity assuring networks, the information (queries, content, etc.), is not transmitted directly between the interested nodes, but through a series of intermediary peers, thus abstracting the connection between sender and receiver, at the expense of overall efficiency.

#### *2.2.8.3.3.1 Relevant Proposed Schemes*

Some of the most noteworthy anonymity assuring protocols proposed, are presented next. The strategies employed by such protocols, to provide for peer anonymity generally belong to one off the following types [2]:

- Disassociation of Content Source and Requestor – This strategy is based in the relaying of information by intermediary nodes, between content source and requestor, so that they can remain unknown to each other and other nodes. Some practical examples are:

    - Freenet [10] – Freenet defines (and implements), its specific anonymity providing protocol, which is based on this approach.
      Its main objective is to protect the anonymity of requestors and inserters of content, and also to preserve the privacy of content storing nodes [10], and users, by rendering impracticable the discovery of the initial origin or destination of any piece of information, found traversing the network.
      Freenet's protocol is based on a mechanism which determines that a successfully discovered resource is transmitted to the requesting node not directly but instead through every node that forwarded the original search request. Also, any peer along the content retrieval path can present itself, or another randomly selected node, as the data source. Thus, no association exists between the content requestor and the content sending peer.
      Furthermore, the hops-to-live counter value, initially set for search messages, is arbitrarily chosen, so as to impede the determining of the distance of the message from the originator peer [2].
      With the execution of more and more searches, each peer will "perfect" its routing table which will render the search more efficient;

    - Return Address Spoofing – This is a practice which can also be used to hide the identity of data sending peer (query or content). It is based on the tampering the headers of IP messages. Every IP packet carries in itself the IP address of the source computer. Given that this information is not needed by the routers, it may be altered without harming the message's routing through the network. That information is only necessary in the case of TCP for control reasons, but UDP can dispense it entirely. Therefore if the UDP protocol is used, and a random return address is inserted, a sender can thus distribute

data and hide its identity from other nodes. Still, given that this practice is associated with unlawful or illegitimate activities, it is frequently prohibited by ISPs;

- The broadcasting of information can also be used for the provision of receiver anonymity, by multiplying the number of recipients and, thus, “diluting” the real receiver in a population of false ones.

- Anonymous Connection Layers – Such schemes employ sub-infrastructures, within the overall network, in order to establish a specialized layer(s) of nodes for the provision of anonymity [2]. Examples of such protocols are:

  - Onion Routing [62] – This is a general purpose protocol for the provisioning of anonymous connectivity over public networks. The main requirement of this system is that all sender nodes know the public keys of all the other peers. All the messages exchanged in the network are randomly relayed through a layer of peers called “Core Onion Routers” (CORs). Whenever a connection is to be established, the initiating peer chooses an arbitrary path through the CORs, and creates a recursively layered and encrypted data structure carrying the necessary routing information called “onion”. Each of the layers of the onion is ciphered with the public key of the corresponding COR node. When such a node is to forward some information and, thus, is given the routing onion, it unwraps a layer (its corresponding layer), decrypting it with its private key. This way the COR node obtains the identity of the next COR node in the path and a new onion (with one less layer), to forward to that router. Given that the inner layers are ciphered with other node’s keys, no COR node has a full knowledge the path. Each router knows only the identity of the next and previous ones;

  - Mix networks [60] [63] – This scheme provides anonymity by forwarding messages from node to node. The specific characteristic of it, is that it does not relay each message as it arrives. The nodes first gather a number of messages and only then do they forward them, mixed up. Such a procedure can effectively hide the sender and the receiver of a transacted portion of information, from an attacker with full network surveillance capabilities. The main disadvantage, of this scheme, is that the message accumulation phase is difficult to implement, or at least introduces inefficiency in content distribution P2P systems, due to their heterogeneousness and inconstancy;

  - In [64], a system providing user, server and active-server document anonymity for a P2P infrastructure, is presented. The system presupposes the existence of a public key infrastructure, so any node can know any other peer's public key. In this mechanism the peers can fulfil four different roles: publisher, forwarder, storer and client. When a publisher wishes to distribute a document, it splits it into encrypted shares and uses an anonymizing layer of nodes (the forwarders), to determine the nodes which will store the shares (storers). The way in which the content is transported, and delivered, is similar to the onion routing protocol;

  - Crowds [61] – This protocol adopts the strategy of “blending into a crowd" for the provision of anonymity in web transactions. It groups peers into a large, and geographically distributed, set (crowd), which collectively performs requests on behalf of its constituting nodes. Each message that is to be sent to some server, or peers, is randomly routed through the crowd until one of the crowd’s peers decides to forward it to its destiny. In this way neither the

receiver, nor the forwarding peers in the crowd, know who the message's originating peer was;

- Tarzan [65] – This is a decentralized anonymizing protocol based on a network layer infrastructure, which provides anonymous IP tunnels between peers. Its operation is based on the routing of packets through tunnels of arbitrarily chosen Tarzan nodes, using mix-style layered encryption [2].

- Censorship Resistant Lookup – Such schemes provide anonymity as a means to assure censorship resistance. They do so by pursuing some specific goals [66]:

  - Data may be inserted into the system, without revealing the identity of the inserter. This way, attacking those who insert information will not be an effective means of censoring it;

  - Data may be retrieved from the system without revealing the identity of the recipient. This way, targeting those who request information will not be an effective means of censoring it;

  - It must be improbable or unfeasible to make a node responsible for a specific document, so that assuming responsibility for a resource and then destroying it will not be an effective means of censoring it;

  - The association between content, and the nodes which store it, must be difficult. This way identifying and attacking individual container nodes will not be an effective deterrent of the diffusion of those contents.

In [66] a censorship proof mechanism, based on the Chord lookup service, is presented. It provides publisher, storer and retriever anonymity, and it makes it harder for a peer to deliberately take responsibility for any specific resource, under distribution in the system.

#### *2.2.8.4 Content Security*

The purpose of content security provisions is to enable the peers, in a P2P system, to be able to validate the integrity and authenticity of exchanged content. Furthermore, such provisions frequently also include tools to enable the assurance of content availability.

##### *2.2.8.4.1 Content Integrity*

In P2P systems, data integrity validation is typically assured though the employment of signatures or checksums. In such schemes (and in the context of content data integrity), at content insertion time, a checksum or signature of the content is calculated. Upon content retrieval, every node needs to recalculate that value, over the retrieved information object, and check it against the original one. Content integrity validation enabling schemes vary predominantly in the manner through which the content file and its checksum/signature are associated to each other. Some relevant such schemes are the following:

- Data detached certification – in these schemes the content is distributed independently from its security signature. Upon (or prior to), content retrieval, the retrieving node obtains the value of the checksum from the content indexing node (structured networks), the content storing node (typically, in unstructured networks), or from some other provision involved in the process of content discovery and/or location (e.g. from the .torrent file in BitTorrent). It then proceeds to validate it against the one it calculates over the retrieved content. These schemes, though, generally

depend on the reliability of the inter-peer communication channels and on the honesty of peers, or on some external identification providing infrastructure (i.e. PKI);

- Data Self-certification – in this mechanism (employed in structured networks), whenever a peer wishes to make a specific content available for distribution in the network, it computes a cryptographic hash of that content, using a universally known (throughout the network), hashing algorithm, and thus produces the content's specific location key. When the content is retrieved, from the network, resorting to its specific location key, the retrieving node performs the same hashing algorithm, on the content, to check if the resulting value is the same as the location key, and, thus, to validate the data's integrity. A general proposal for such a mechanism can be found at [67];

#### *2.2.8.4.2 Content Integrity and Availability*

Several proposals exist for schemes which mean to assure both content integrity as well as availability. Some of the most relevant are:

- Shamir's Secret Sharing Scheme based – the algorithm described in [70] is frequently employed in provisions meant to assure the availability and integrity of distributed content. It is based on the encryption of content with a key, which is split into several portions that are distributed between several servers or peers. The key can be recreated from any (sufficiently big), subset of portions of the original key. The content is distributed together with a portion of the key. For the content to become unavailable a large number of the servers/peers must fail;

- Anonymous Cryptographic Relays – this scheme [64] is based on the abstraction of the connection between the final user and the content storing nodes, through the anonymity of his location, in order to make the system the least subjectable possible to censorship. According to this scheme's operation mode (which depends on the assistance of a PKI), a publisher first chooses several forwarding nodes to which he sends (via anonymous connections), fractions of the encrypted resource. These nodes, in their turn, will select other peers, to play the role of content storers, and hand them (anonymously), the data to be stored. After this phase, the publisher announces the content and the associated forwarders. For a client to have access to content, it will have to access the forwarders and retrieve, through their efforts, the content saved at the storers;

- Erasure coding – this mechanism is similar to the information dispersal algorithm. The difference is that the distributed portions of information are named using a secure hash over the object contents, providing these portions of data with unique identifiers, which contributes to a better assurance of data integrity.

#### *2.2.8.4.3 Content Authenticity*

The existing distributed schemes to assure content authenticity may be divided into the following categories [69]:

- Oldest document based – In this type of scheme, the first object to be submitted with a specific identification is considered to be an authentic match for a query looking for that specific identification. Time-stamping systems as the one present in [68] can be employed in the implementation of such schemes;

- Expert based – In this approach, an object is deemed authentic by an "expert" or authoritative node N, which keeps track of signatures for all objects/files ever

authored by any user of N. If at any time it is necessary to verify the authenticity of a file, supposedly, authored by any of N's users, node N can be consulted. If and when node N is unavailable the scheme fails to function;

- Voting based – To deal with the possible failure of N, another possible authenticity assuring scheme, takes into account the "votes" of many experts. The expert nodes may be operated by human experts or regular users which simply vote on the authenticity of a file by deciding to store a copy of it or not. This scheme may be augmented through the weighing of experts' opinions by their reputations (as maintained by such reputation systems as those presented in [34] [35]). The key issues with this scheme are how to prevent spoofing of votes, of nodes, and of files.

## 2.3 Noteworthy Systems

### *2.3.1 Introduction*

Since the rise of Napster, (the first notorious P2P network), P2P technology has endured a prolific evolution (mostly in a non-commercial context), which has resulted in the technical panorama described in section 2.2. As such, numerous P2P protocols and systems coexist in today's Internet.

An exhaustive description of all such systems would be both unnecessary and practically unfeasible for a presentation of the present scenario in the field of P2P content delivery.
The exposition that follows, will focus on some the most noteworthy, (either due to their popularity or to their technical relevance), P2P networks in existence. These will be Gnutella, BitTorrent, eDonkey and FastTrack [71].

It should be noted that, for many of these systems and applications, no exhaustive formal source of information exists, and, as such, information, must, on many occasions, be collected from multiple secondary sources, such as, scientific articles, Web sites, message boards, forums, etc. In the following sections the presented information, was in many instances obtained through this process.

### *2.3.2 Gnutella*

Gnutella began as a fully decentralized protocol for distributed search on a flat topology of peers [15]. This leads to a lack of scalability in the search mechanisms which undermined the overall performance of the network. Gnutella has thus evolved into becoming a partially centralized system, which employs an overlay network. This system's nodes are either leaf nodes or higher level nodes, called ultra-peers. Ultra-peers are high capacity nodes which behave as proxies for the network's leaf nodes.

When a node initially connects to the system it must discover the location, and connect to, at least one other node, of the ultra-peer type. Different procedures have been developed for this purpose, such as, shipping a list of addresses of working nodes with the software, employing updated web caches of known nodes (called GWebCaches), etc. [15].

Once connected to an ultra-peer, a Gnutella node requests an updated list of addresses of working peers with which it will interact. If it is a leaf node it then proceeds to publish its contents list to a number of known ultra-peers.

Peers interact through the exchange of messages over TCP/IP. Each issued message possesses a 128 bit [47] unique and randomly generated identifier and all peers maintain a short term memory of the recently routed messages. This memory is used to prevent the re-transmission and to implement back-propagation of messages. All messages have a

“TimeToLive” (TTL) and a ”HopsPassed” fields, which determine their lifespan within the system.

The Gnutella protocol does not provide the system with any control over its topology or over content placement [15]. It is therefore an unstructured network. Thus, in order to locate a specific data item, a peer queries its neighbours.

The initial data discovery method in Gnutella was flooding. The query was forwarded to all neighbours of the inquiring peer within a certain radius. If a peer, receiving a query, knew the location of the desired content, it would send back a response to the originating peer via the query’s incoming path. If it did not know the location of the content it would forward the query to its neighbourhood [17]. The described content location mechanism is highly resilient to peer population transience. Still it is not scalable and contributes to an overburdening of the network, especially in an architecture without ultra-peers.

Later versions of the protocol and its peer implementations have attempted to deal with the system’s lack of scalability. Some changes were introduced such as:

- search results are returned over UDP directly to the node which initiated a search. This diminishes the amount of traffic routed through the Gnutella network, increasing its scalability;
- a number of techniques were adopted in order to reduce traffic overhead and make searches more efficient. The most relevant of these were QRP (Query Routing Protocol) and DQ (Dynamic Querying). When QRP is employed, a search arrives only at those peers who are likely to possess the desired content, this way the searches for rare data items grow in a more efficient way. When DQ is employed the search stops as soon as enough search results have been acquired. This greatly diminishes the amount of traffic caused by searches for popular content [72].

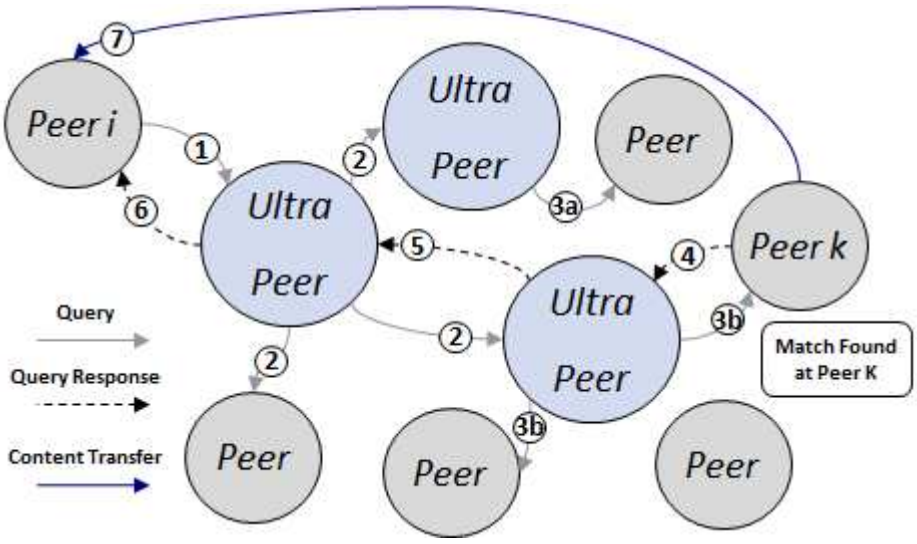


Figure 6 – Gnutella Structure and Content Search

The, already mentioned, introduction of a layer of ultra-peers also contributed greatly for the increasing of the system’s scalability. In the new architecture, content discovery may be performed as an optimized “flooding” that occurs only in the Ultra-peer sub-structure as depicted in Figure 6.

In the Gnutella network, content transfer is performed over HTTP, in a direct connection between the content supplier and content requester. The retrieving node establishes a connection to the serving host and makes a standard HTTP request for the desired content.

If the serving peer is behind a firewall the procedure is altered. The serving peer is instructed to push the content to the content requiring peer. The protocol also supports simultaneous downloading from multiple sources [73].

The strategy employed by Gnutella for content replication throughout the system is that of passive replication. Only the peers who download a specific content become its distributors.

The Gnutella system has no provisions or incentive mechanisms for the enforcement of fair use, on the part of the constituting peers. Requests are to be served in order of arrival, for as long as there are upload slots available at the serving peer. If a specific implementation (peer) is to behave in a systemically damaging selfish way, the system is not able to prevent such behaviour [73].

Gnutella allows for the checking of the validity of retrieved content, but only after download completion. This means that a corrupted chunk of the retrieved file will imply the downloading of the entire content again [73].

### *2.3.3 BitTorrent*

BitTorrent [74] is a centralized P2P system. It employs central nodes called “trackers” for the managing of users' downloads. A tracker is contacted whenever a network node wishes to download a file. Trackers maintain and update an index of torrents. That is, they keep track of all the nodes which at a specific moment are participating in the downloading of each specific file (both partially and completely). They thus handle the facilitation of the connection between peers for the downloading and uploading of content.

The main BitTorrent architecture is an unstructured one. Still with the evolution of its implementations (peers), trackerless torrents are now also supported through means which render the resulting system a structured one. Trackerless operation is supported through the employment of a Distributed Hash Table (DHT). This is a layer which is added on the top of the BitTorrent network. The strict BitTorrent network and its DHT enabled portion operate independently. Each DHT enabled node is responsible for indexing a certain percentage of hash files on the network thus building a distributed indexing system [73], where each peer behaves as a tracker.

Azureus [75] [76], and the official BitTorrent client have been the first to add a DHT layer. Both employ the Kademlia [77] structured overlay, but the two resulting networks are not compatible [73].

For content retrieval, BitTorrent supports the simultaneous downloading from multiple sources of fractions of the same content, and also the sharing of partially downloaded files. BitTorrent peers can thus upload a file while still downloading it.

The BitTorrent system divides files into fragments of fixed size (256 Kbytes), and the content of each peer is tracked on a block basis [17].

A peer distributing a data file, handles the file as a number of identically-sized blocks. The peer calculates a checksum for each of the blocks, using the SHA1 hashing algorithm. The calculated hashes are stored in a .torrent file. When another node, later retrieves a particular fragment, its checksum is computed and compared to the recorded one to test the fragment’s validity. A peer which provides an entire file is called a seeder, and that which delivers the initial copy is the initial seeder [74].

Torrent files are composed by:

- "announce" section, specifying the URL of the tracker which is responsible for the tracking of the specific content's diffusion throughout the system, that is, the Torrent;

- "info" section, containing the file's names, its length, the fragment size used, and a SHA-1 hash code for each of the fragments

Torrent files are generally made available online (via websites for instance), and registered with a tracker (the tracker specified inside). The selected tracker maintains updated lists of the peers currently involved in the torrent.

For content retrieval to take place users must somehow (browsing the web) obtain the torrent information for the desired content (the .torrent file). After it is downloaded and supplied to a BitTorrent client, it connects to the tracker(s) indicated in the torrent file. The tracker(s) delivers to the BitTorrent client a list of peers which are at that time transferring fractions of the file(s) specified in the torrent. The peer then proceeds to connect to those nodes to retrieve the various pieces. This content retrieval procedure is depicted in Figure 7.
The group of peers participating in a torrent is called a swarm. If only the initial seeder is included in the swarm, further peers will directly connect to it and request pieces of the fragment from it. As the swarm increases, the new peers begin trading fragments with one another, instead of downloading them from the seeder.

This digital content distribution system employs a strict reciprocity strategy for content diffusion. That way, a peer responds with the same action that its other collaborating peer performed previously. They upload to those peers who have uploaded to them and they download from those which have downloaded from them. This bartering scheme is a trade-based incentive mechanism designed to discourage free-riding, by making peers select other peers from which data has been previously received. Nodes with high upload capacity will most likely also be able to download content with a high speed and the average download capacity for a peer will be reduced if its upload speed is limited [17].

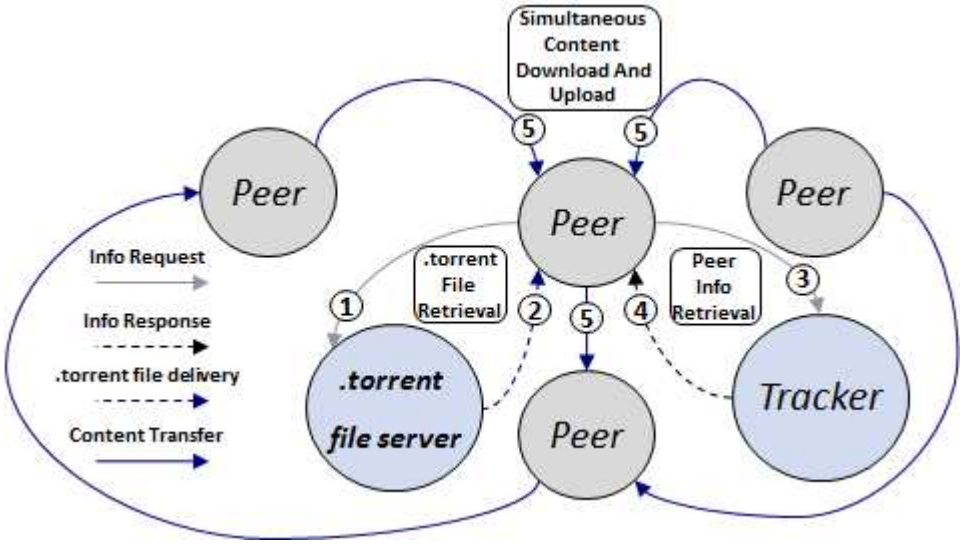


Figure 7 – BitTorrent Protocol Operation

The bartering procedure also contributes to a better spreading of content among the peers thus improving reliability [17]. Given the implications of the reciprocity strategy employed by BitTorrent, its content replication scheme is quite efficient as it forces peers to participate in it for their own interests. Nonetheless it may be considered to be a form of passive content replication.

A strict bartering policy may nonetheless have negative effects on the system (for instance newly joined peers would in many cases be unable to receive data because they do not possess any fragments yet to trade themselves). To compensate for such negative effects the protocol employs an "optimistic unchoking" mechanism. According to it, peers reserve a portion of their available bandwidth for sending content fragments to random peers (not only preferential partners, but also others). This allows for the discovery of more advantageous partners but also ensures that new nodes on the network have the opportunity to join in [74].

Choking is a temporary refusal from a peer to upload content. At that time content downloading may still occur and the connection does not need to be renegotiated when the choking terminates. A peer may perform choking for several reasons, such as to get a better congestion control or performing a more advantageous bartering scheme for content trading [17].
For optimal efficiency the choking algorithm should [17]:

- reduce the number of simultaneous uploads for good TCP performance;
- avoid fibrillation, which is a rapid choking and unchoking;
- supply the download service to peers who supply (are supplying) it back;
- periodically experiment unused connections to determine their possible usefulness.

The BitTorrent protocol (through the official peer implementation) avoids fibrillation by only changing the chocked peer every ten seconds. It performs reciprocation and limits the number of uploads by unchoking the four peers with the best download rates.
If a downloading peer possesses a complete file, it uses its upload rate rather than its download rate to decide which peer to unchoke. Peers which are optimistically unchoked change every 30 seconds [17].

BitTorrent employs "Info hashes" for the identification of files or portions of data. The .torrent files carry a list of such block hashes in order for the peers to be able to verify that the individual downloaded blocks of a file are not corrupted. Corrupted blocks must be re-downloaded [73].

### *2.3.4 eDonkey*

MetaMachine Inc. was the original developer of the eDonkey protocol and peer, the eDonkey 2000 (or ED2K). Still, their software was reverse engineered and several other clients were developed. In 2005, due to legal pressure from RIAA, MetaMachine discontinued the distribution of their software [78] [79].

This change, however, has had little impact on the overall network's operationally. Alternative peers, such as the open source eMule, long before the distribution of the eD2K client was ceased, had already become common place in the network, and have continued operating and thus implementing, the eDonkey network.

The network relies on highly capable central servers which are loosely connected and are run by users and the "community". The servers perform the role of communication hubs and content indexes, allowing users to locate files within the network. Given the fact that the network is composed by two functionally different types of peers, (client and server), it is thus a hybrid decentralized P2P system.

The central servers can be subject to heavy traffic and, thus are more vulnerable to attacks and constitute a potential bottleneck. To deal with this issue, the Overnet protocol was developed as an evolution of eDonkey. It employed a Kademlia [77] resource placement scheme. This Kademlia enabled part of the network is thus a structured decentralized P2P system.

To initially connect to the network, a peer (client) must have *a priori* knowledge of the IP address, and port, of at least one other peer (of the server type) in the network. After connecting, a peer must register the content files, that it possesses, and that it is able of sharing, by providing the meta-data describing the files, to the server peer [17].

Content files, within the eDonkey system, are divided into blocks. A checksum is computed for each of the blocks and they are propagated between clients on demand. A checksum of all the checksums is employed by the system as the file identifier [80]. During content retrieval time, if a corrupted block is detected it is discharged and retransmitted [73].

Content may be located either by querying the servers with values that will be matched to the descriptive meta-data that the servers host, or by requesting a particular file through its unique network identifier. Servers deliver, to clients, the locations where the desired files may be obtained. The files are directly exchanged between client peers [17]. EDonkey queries support fields such as name, size, extension and bitrate [80]. Clients peers may also browse the listings of contents of other client peers [80].

The eDonkey network (the set of eD2K peers, more specifically), uses Multisource File Transmission Protocol (MFTP), to enable the simultaneous downloading from multiple sources, and the sharing of partially downloaded files. An eDonkey peer can thus upload a file while still downloading it [73]. This fact agilizes the overall distribution of large files.

In what regards the enforcement of fair use, in the eDonkey system, the employed measures vary with the implementation of the protocol (peer). The official peer (ED2K) uses a basic queue system to attend to requests and employs the proprietary Horde system. Shareaza permits the users to specify their own criteria. The eMule's peers employ a credit system where the more a peer uploads to another peer, the faster the first peer advances in the second's waiting queue [73].

### *2.3.5 FastTrack*

FastTrack [81] is a proprietary protocol. It appeared at the end of the first generation of P2P networks in 2001. In time it has forked into a number of different and mutually incompatible networks.

The FastTrack protocol defines a semi-centralized P2P network. Its constituting peers fall under the category of ordinary nodes (ONs) or supernodes (SNs). The SNs occupy a more central position in the network, and handle a greater and more relevant set of tasks. They perform the role of temporary index servers. Any of the FastTrack's nodes, with sufficient CPU capability and network connectivity, can become an SN. These central nodes are elected by the system through a decentralized process [73]. The fulfilment of the role is voluntary.

When a FastTrack ON application is initiated it selects a parent SN, with which it establishes a semi-permanent TCP connection. The procedures employed, for the ON to know where to connect, are somewhat similar to those used by Gnutella.

The FastTrack system is unstructured. Content discovery is performed through the employment of query diffusion. The SNs (also called super-peers) store the metadata about the contents, shared by the ONs under their responsibility, in an index. When a node wishes to obtain a specific content, a query is issued to its respective SN, containing the user defined keywords. After that, a, Gnutella like, broadcast based search is performed in a highly pruned overlay network of SNs. For each match at an SN database, the respective SN returns the IP address, server port number, and metadata corresponding to the match.

Figure 8 presents a view of FastTracks's architecture.

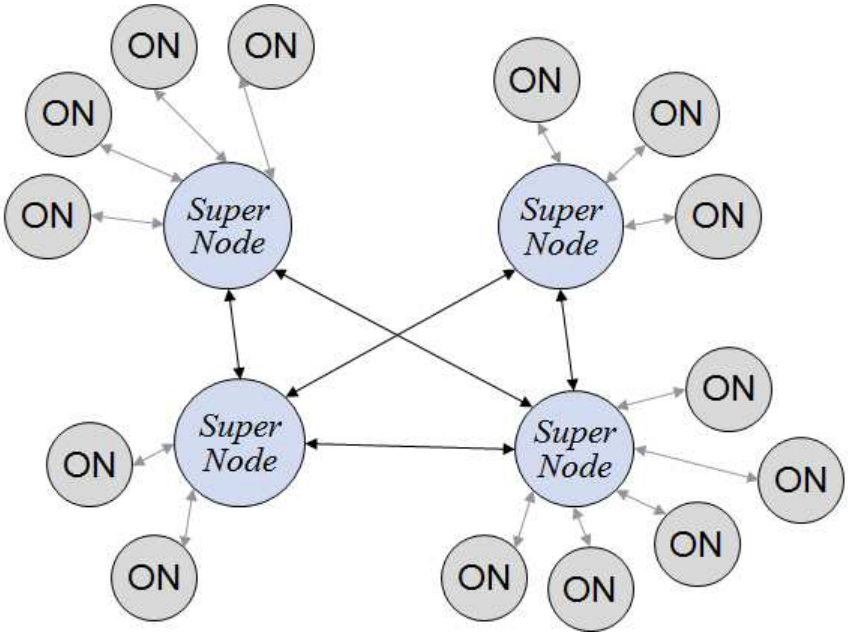


Figure 8 – FastTrack Architecture

FastTrack can still operate without the presence of SNs but its performance is degraded. The maintaining of the SNs' indexes, and the broadcasting of queries between them, are, nonetheless bandwidth consuming [17] strategies.

The indexes stored at the SNs are built from information received from the ONs. The information about the content files, that an ON is harbouring, which it shares with the SN includes [82]:

- the file name;
- the file size;
- the ContentHash – a hash is calculated for every file, and it is used as its signature. The ContentHash is the only identifier used to identify a file in an HTTP download request. If a download, from a specific peer fails, the Content-Hash allows a FastTrack peer to automatically search for the specific file, without having to emit a new keyword query;
- file descriptors (such as artist name, album name, and text entered by users). These are used for keyword matches during querying.

The TCP traffic that FastTrack peers exchange, serves four main functional purposes [82]:

- Signaling traffic – includes handshaking traffic for the establishment of connections between nodes, metadata uploaded from ONs to SNs, supernode lists and queries and replies. Signaling traffic is encrypted;
- File transfer traffic (e.g., MP3s, videos, etc.) – takes place directly between the end peers without passing through intermediate SNs. File transfers are not encrypted and are sent within HTTP messages;
- Commercial advertisements – sent over HTTP;
- Instant messaging traffic – encoded as Base64.

This protocol supports multi-source downloading, but not partial sharing of files [73].
The specific implementation of this protocol, done by the KaZaA client, employs unencrypted HTTP transfers for content retrieval. And all transfers, when using this peer technology, include KaZaA-specific HTTP headers [17].

The strategy employed by FastTrack for content replication throughout the system is similar to that employed by the Gnutella protocol.

The Kazaa variant, of the FastTrack system, implements an incentives mechanism to reward fair use of the network's resources. The mechanism involves the evaluation of the "Participation Level" of peers, which relies only on a locally computed value. The formula used for its calculation, is proportional to the ratio of uploaded megabytes of data to the network, by a peer, over the downloaded megabytes from the network, by a peer. A maximum value for such level is also defined. The participation level value is sent to each peer that a given node wants to download from, where it determines its position in the remote node's download queue.

As the value of the participation level is calculated, and stored, locally, the incentives system defined by FastTrack is, obviously, of little robustness. It was eventually cracked and some clients were developed which present a modified participation level, which is constantly set at the maximum value [73].

The FastTrack protocol is not very resilient to polluting. This is so because the employed hashing algorithm performs only incomplete file hashes [73], and no other system wide provision exist to assure content validity. The hashing protocol has, nonetheless, been subjected to some recent improvements.

## 2.4 Summary

As exposed in the previous sections, the technical landscape of P2P content distribution is a very diverse one. It encompasses a vast group of tools and proposals for a varied set of different purposes.

The first notorious P2P system, of relatively recent times, was Napster. This system had a hybrid decentralized architecture (as explained in section 2.2.2 of the present chapter), and thus, a central coordinating entity. This entity granted it a good operational performance, but it was also a legally targetable point. Napster was, thus, targeted with lawsuits and, eventually, shut down.

P2P systems then progressed in the direction of increasing decentralization. This was triggered by the desire to make P2P systems more independent and resilient to node failure and attack, but also, to eliminate provisions, of such systems, that where legally targetable (the central or coordinating ones). This evolution, however, brought scalability and islanding problems. Thus, in time, a return as occurred to more centrally coordinated modes of operation, (BitTorrent, FastTrack), in order to increase operational efficiency.

Globally, the available technologies, for P2P content distribution, provide more or less robust means for:

- In the context of operational reliability:
    - relatively robust content location – structured systems generally provide the means for a relatively efficient location, once a content object's specific ID is known;
    - efficient content exchange – as peers, typically, exchange content directly, they eliminate the need for any central repository, and distribute the content delivery workload though the peer tissue. Protocols which enable a fragmented content exchange are more efficient as they enable the immediate discarding of corrupted fragments. Furthermore, protocols (e.g.

BitTorrent), which have some form of coordination of the content exchange process are more efficient that those which do not;

- relatively high resilience to the topological transience which characterizes P2P networks in the context of content availability – unstructured networks have the best performance, in this regard, as in such systems there is no mapping of content objects to peer. If some peer goes offline, its content may, generally, be obtained from other peers;

- some effective fair use enforcement, through bartering – this is achieved by BitTorrent [40];

- In the context of security:
  - some content integrity validation – most of the predominant P2P systems enable the validation of the exchanged content objects' integrity. These provisions are however of relative robustness as they generally depend on the safety of the communication channel, between peers, through which the content checksum is exchanged;

However, the defined technologies, protocols and platforms, typical lack, or presuppose the absence, of any centrally coordinating or absolutely trusted provision. There is thus no point of "high ground", in such systems, which can: maintain a coherent global vision of the system; enforce some form of globally aware coordination or resource management; or be the source of ultimately and absolutely trustable information.

Given the transience, and the inherent unreliability, of the tissue that composes P2P networks, the above mentioned characteristic, of the currently predominant P2P technologies, is at the root of most of said technology's shortcomings. These are;

- In the context of operational reliability:

  - content discovery and location – as, typically, no entity exists with a global and coherent view of the entire content object repository, the discovery and location of such content requires the execution of distributed algorithms. In unstructured networks said discovery is performed either by query diffusion though the network's peers or by inquiring a super peer. The earlier alternative is inefficient, unscalable and prone to flooding the network. The latter is more scalable but still does not enable a search over the content in its entirety. In structured networks, content location is fairly robust if the exact identifier of the target content is known. This however means that said identifier must first be, somehow discovered. This is a pending problem of these networks as their supporting of keyword based searches is, at best, very fragile. Furthermore the coherence of the mapping of resources to peers, and consequently, content availability, is difficult to maintain, given the transience of the peer collective and the existence of malicious peers;

  - content exchange/retrieval – the way that content is distributed/diffused across P2P systems is generally efficient, in the sense that its exchange depends on no central server and the content distribution workload is diluted across the entire peer collective. However, there is still room for improvement in terms of the coordination of the exchanges, that is, in terms of determining which peers retrieve which object from which other peer and making sure that they all contribute. In this regard the BitTorrent protocol presents a higher efficiency. Nonetheless, if, in a P2P system, a global overseeing entity exists, it may enforce global content diffusion patterns, throughout said system, with optimal efficiency;

- connectivity of the system's tissue and content availability – given the decentralized and transient nature of P2P networks, the occurrence of islanding is frequent, especially in completely decentralized (with no super nodes) systems. This consists of sections of a same system (e.g. Gnutella), loosing contact with each other. This leads to the systemic behavioural variability and inconstant content availability;

- content management - most P2P systems present little content management capabilities (e.g. deletion, update, validity expiration, versioning, etc) or none at all. When such capabilities are present they are generally not very reliable given the lack of overseeing provisions to guarantee their adequate enforcement;

- In the context of security – all shortcomings in this field, ultimately, result from the typical lack of a central trusted entity. These basically consist of:

  - insecure identification of peers and users – the predominant proposals for the identification of peers and users, in P2P systems, avoid resorting to any central trust providing authority and, instead, employ distributed, frequently quorum based, algorithms [49] [50] [51]. These are vulnerable to vast and sophisticated attacks and imply a considerable operational overhead. These mechanisms are therefore not really reliable;

  - insecure communication – in line with the reality exposed in the above bullet, the predominant mechanisms for assuring communicational integrity authenticity and un-deniability are fragile at best. In terms of anonymity assurance, however, some of the developed schemes present some satisfactory robustness;

  - fragile media object integrity and authenticity assurance – the few integrity assuring measures enforced by the predominant P2P systems in operation, are very fragile as they depend on the security of the communication channel, (frequently inexistent) and on the honesty of peers. Authenticity assuring mechanisms are even more fragile and subjective;

  - generally inefficient enforcement of fair use – fair use is enforced either through the employment of peer reputation maintenance mechanisms, or through trade based mechanisms. The latter can further be divided into those which are resource trading based and those which are micro payment based. Resource trading based (or barter based) mechanisms are the most efficient and less prone to fraud. These, nonetheless, present some issues as they can have a negative impact on the efficiency and flexibility of content exchange, given their frequently rigid demands for reciprocity;

  - virtually absent rights management provisions.

In light of the content above exposed, it can be concluded that the weak spot of P2P technologies are its security related aspects.

# 3 DRM Technology Survey

## 3.1 Introduction

The purpose of Digital Rights Management (DRM) solutions is to provide content producers/distributors with the capability to govern and administrate the usage of their information goods, throughout the entire on-line digital value chain. DRM is composed by all the activities and procedures involved in the electronic and informatics related management and marketing of usage rights over digital content.

DRM tools may be embedded in physical support backed media as well as in digital goods which are distributed online, such as music files, e-books, games, videos, etc. The development of DRM solutions has been underway for some time and some solutions and practical implementations have been put forward. Still, this technological field has been characterized by some relatively difficult progress for technical reasons as well as for social, legal and economical ones. Thus, as a whole the field is yet in a state of maturing, and the existing systems are mostly proprietary and are generally unable to inter-operate, what many times renders their usage uncomfortable to consumers.

The possibilities of DRM application, the scope of DRM concepts and their global implications, (technical, legal, etc), are very vast, rendering the elaboration of an all-encompassing study unpractical. This overview will focus on the technological aspects of DRM, that is, on the management of usage rights over digital goods by control systems and on the most relevant of such systems.

## 3.2 Technological Description

DRM systems consist of a set of information technology components and services, supported upon copyright related law, whose aim is to provide the necessary tools for the creation of a controlled environment for digital goods distribution and consumption.

The core operation of on-line DRM technology consists of protecting and facilitating the selling of usage licenses for digital content. Such licenses are portions of digital data that express specific usage rules and rights over digital content. These rules can be crafted so as to implement various business models, such as rental, subscription, pay-per-use, etc [83].

DRM technology is generally divided into the two following operational areas [84]:

- rights expression – includes the identification and expression of the rights pertaining to works and to the parties involved in their creation, distribution and consumption;
- rights enforcement – includes the technical enforcement of usage restrictions due to the rights in context.

The first operational area is generally dealt with through the employment of metadata tools. The second one is handled through the usage of encryption schemes and software or hardware provisions.

### *3.2.1 Main Logical Mechanisms*

#### *3.2.1.1 Content Identification*

Identification is the process by which a unique universal identifier is linked to a specific media object, allowing its unambiguous distinction from all its peers. If identifiers are not universal, their uniqueness and meaningfulness will only be assured within the context of their corresponding namespace (its naming authority and all entities which can interpret the

naming protocol) [84]. For instance, the uniqueness of id “ISBN-13: 978-1587052576”, is assured by the entity managing the International Standard Book Number namespace.
Some of the most relevant content identification standards are presented below in Figure 9 [85].

| Identification System | Standard, Scope and Agency or working Group |
|---|---|
| **Book Item and Component Identifier (BICI)[90]** | • ANSI/NISO Draft Standard for Trial Use<br>• Books |
| **Digital Object Identifier (DOI)[91]** | • ANSI/NISO Z39.84-2000 (syntax only)<br>• Any IP entity<br>• International DOI Foundation (IDF) |
| **International Standard Audiovisual Number (ISAN)[89]** | • ISO/DIS 15706.2<br>• Audiovisual abstractions<br>• ISO/TC 46/SC 9 Working Group 1 |
| **International Standard Book Number (ISBN)[92]** | • ISO 2108:1992<br>• Books, software, mixed media, etc.<br>• The International ISBN Agency |
| **International Standard Music Number (ISMN)[92]** | • ISO 10957:1993<br>• Printed music<br>• The International ISMN Agency |
| **International Standard Musical Work Code (ISWC)[93]** | • ISO 15707:2001<br>• Compositions<br>• The International ISWC Agency, International Confederation of Societies of Authors and Composers, International Standard Recording Code (ISRC) |
| **International Standard Recording Code (ISRC)[94]** | • ISO 3901:1986<br>• Audio and video recordings<br>• International Federation of the Phonographic Industry |
| **International Standard Serial Number (ISSN)[95]** | • ISO 3297:1998<br>• Serial publications<br>• ISSN International Centre |
| **International Standard Textual Work Code (ISTC)[96]** | • ISO/WD 21047<br>• Textual abstractions<br>• ISO/TC 46/SC 9 Working Group 3 |
| **Publisher Item Identifier (PII)** | • Elsevier Science Ltd.<br>• Textual abstractions<br>• Elsevier Science Ltd. |
| **Serial Item and Contribution Identifier (SICI)[97]** | • ANSI/NISO Z39.56-1996 (Version 2)<br>• Components of serials |
| **Unique Material Identifier (UMID)—also known as “Universal Media Identifier”** | • Society of Motion Picture and Television Engineers (SMPTE) 330M-2000<br>• Digital content<br>• SMPTE Registration Authority, LLC |
| **MPEG-21 DIID[98]** | • ISO/IEC FDIS 21000-3<br>• Digital content<br>• ISO/IEC JTC1/SC 29 WG11 |

Figure 9 – Content Identification Schemes and Standards (data obtained from [85])

### *3.2.1.2 Content Metadata*

Content metadata, in the present context, is information pertaining to digital content. It is, generally, not directly connected to the security provisions of DRM systems. It usually serves auxiliary, or user experience augmenting, purposes.
Such content metadata may carry information about a variety of subjects such as:

- semantic description of media content;
- the product's origin;
- business or commercial exchange related data;
- etc.

Metadata formats are generally intimately connected to the type of content which they were crafted to describe, and thus, are frequently not suited for other types of content.

#### *3.2.1.3 Rights Expression Language*

Rights Expression Languages (RELs) are tools that allow the expression of rights and restraints on the usage of digital goods, in a language that is commonly understandable in some digital context.

RELs are generally required to be:

- Fully expressive – They must enable rights holders (or their proxies) to express their rights and interests in content, as well as to specify contractual agreements related to it, pertaining to a variety of usage and business models;
- Unambiguous – They must be precise in a way that they are interpreted in the exact way that the rights owner intended;
- Machine readable – REL metadata must be interpretable by automated devices;
- Secure – Instances of REL data (licenses) must be of a nature which assures that any tampering can be detected.

XML is employed as a base syntax by most of these languages.

RELs require extremely semantically precise terms so as to provide unambiguous expressions. Natural language is far from precise, and thus, unsuitable. REL languages therefore require the development of specific sets of highly precise terms so that they may express rights and permissions, without ambiguity and in a common understandable way. Rights data dictionaries are the instances which group these necessary terms [84].

Some of the most relevant examples of REL initiatives or standards are the following:

- Open Digital Rights Language [86] (ODRL) REL – this standard is meant for open use and is under cooperative development. Version 1.0 of ODRL merges Nokia's Mobile Rights Voucher (MRV) and Real Networks' Extensible Media Commerce Language (XMCL) [87]. Current version is 2.0. ODRL is subjected to no license requirements and is available in the same model of "open source" software. It is part of the W3C standards [85];
- The Extensible Rights Markup Language [88] (XRML) – an XML-based usage grammar for specifying rights and conditions governing the access to digital content and services (e.g. digital works, Internet-based services, fragments of information such as an email address, etc [89]), which is owned by ContentGuard;
- MPEG-21 Rights Expression Language (REL) (and the corresponding Rights Data Dictionary) – at its basis is XrML. REL contracts bind together one or several principals (entities), a set of rights that are possessed over a digital resource, and the conditions to which those rights are subjected. Furthermore the object of the contract, (the resource), may be a rights expression itself. Thus MPEG-21's REL may be employed to perform the transfer of rights along the chain of digital goods production and consumption [90];
- Creative Commons (CC) [91] – this standard provides a legal structure and expression language which allows the development of a "*some rights reserved*" context for digital resource sharing on the Internet. The goal of the CC initiative is to

foster creativity by building a vast base of material that can be re-used for the production of new digital goods;

- The Extensible Access Control Markup Language(XACML) – it was developed by OASIS [92]. It is an XML based declarative rights expression language and a processing model which describes how to interpret the policies. It defines a core schema, and corresponding namespace, for the specification of authorization policies over data objects [93].

#### *3.2.1.4 User and Device Authentication*

An important functionality in any DRM architecture is User or Device Authentication. User authentication is needed to allow the creation of a connection between the user, the content and the rights the user has over the content.

Device authentication performs the assignment of unique identifiers to content consuming devices. Such identifiers are employed to assert the device's permission to perform the consumption of the digital asset.

Several technical alternatives for device identification exist:

- employment of the MAC or IP address, on Internet connected devices;
- employment of the CPU ID for Intel machines;
- employment of the IMEI number on mobile devices. In these cases the SIM card can in parallel, be used for User identification.

Some relevant initiatives in this area are:

- Palladium (now known as Next-Generation Secure Computing Base) [94], which is a Microsoft developed system, that employs both software and hardware controls to build a "trusted" computing environment;
- Trusted Computing Platform Alliance (TCPA). This is an open alliance, which was constituted for the goal of creating of a new computing platform capable of providing trust in the PC platform. TPCA is currently designated by TCG - Trusted Computing Group [95].

Some of the most relevant alternatives and initiatives for on-line user identification are the following:

- Microsoft Account [96] (formerly Windows Live ID) – this is a, broadly deployed, online authentication system developed by Microsoft. Its single sign-in service permits a user to develop a single set of credentials and use it throughout all sites that support said service;
- OpenID [97] – this is a decentralized, free and open standard managed by the not-for-profit OpenID Foundation [98]. It enables Internet users to authenticate themselves with many different web sites using a single digital identity;
- Security Assertion Markup Language (SAML) [100] – this is an XML based framework, which was developed by the Security Services Technical Committee of OASIS, for the exchange of authentication and authorization information between security domains (between Identity Providers, which produce assertions, and Service Providers which consume assertions);
- Shibboleth [99] – this is an Internet2 Middleware Initiative which has developed a standards based architecture, and an open source implementation (Apache Software License), for a federated identity-based authentication and authorization infrastructure based on OASIS' SAML.

### *3.2.1.5 Event Reporting*

All the steps of a typical eCommerce transaction, and the usages to which the delivered content is subjected, constitute an event. For instance, the online purchase of a musical content and its consumption implies events such as ordering, transfering, listening and paying for content.

Therefore, the notification of these events to some central system entity(s) is useful for such purposes as:

- the monitoring of the usage of copyrighted material;
- the monitoring of technical grandeurs which characterize the system and determine its performance, such as the connectivity between devices of different system actors;
- the collecting of revenues;
- the collecting of statistics.

Typically, on-line content distribution systems can generate two types of notifications (or Event Reports) [101]:

- Object Action Notification (OAN) – carries information pertaining to the actions that an actor of the digital value chain has performed over a digital object;
- System Action Notification (SAN) – carries information pertaining to the actions performed over any system module but not over an object.

A relevant example of a technical platform for Event Reporting is the MPEG-21 Event Reporting (MPEG-21 ER) standard. It delivers the necessary provisions for, amongst other things, Digital Item producers and distributors to be able to estimate or monitor the usage of their products. It provides a standardized structure for “reportable events” to be specified, detected and acted upon. The comprised reportable events may be connected either with the usage of a Digital Item by a Peer, or with the occurrence of Events connected to the Peer itself [102].

The key concepts at the base MPEG-21 ER are the Event Report Request (ER-R) and the Event Report (ER).

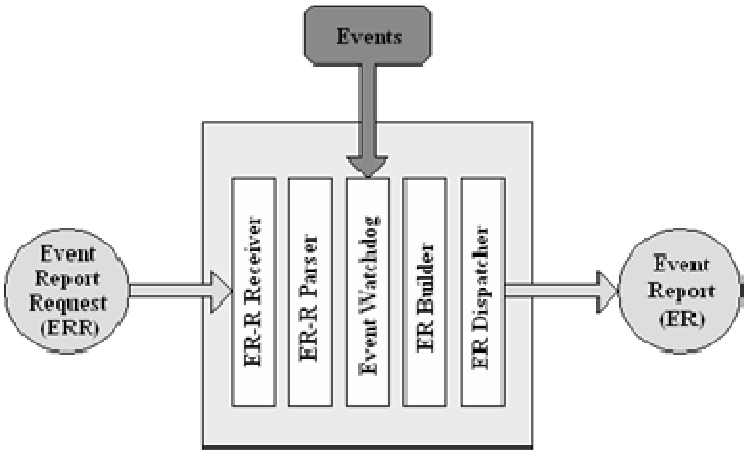


Figure 10 – Model of ER-R Processing and ER Generation (adapted from [102])

The purpose of an ER-R is to define [102]:

- the conditions that must be verified so that the reportable Event is considered to occur;
- the syntax/format of the information which is be supplied, within the payload of the ER, when the reportable Event occurs;

- the intended recipient(s) of the Event Report, (the entities that need to be notified when the reportable Event occurs);
- parameters specific to ER delivery aspects, (such as the transport mechanism and protocol, delivery timing constraints, priority, etc.).

Figure 10 presents a general (non-normative) model which depicts the functional characteristics of ER-R handling and the related generation of ER's.

Within the defined model, the operation begins with the delivery of an ER-R to an MPEG-21 Peer. The ER-R is processed and the Peer waits for Events to "occur". When such an event occurs, it is detected by the Event Watchdog module, which determines if the Event should be reported on, or not, according to the ER-R defined parameters. If all of the Event conditions defined in an ER-R are verified, the Peer builds an ER carrying the data fields specified in the corresponding ER-R. Finally the generated ER is sent, by the Peer, to the ER-R's defined recipient Peer(s) [102].

#### *3.2.1.6 Content Protection*

Content protection is a key element of a DRM system, as it provides the technical support basis for content accessing denial and for the enforcing of content usage restrictions.

Content protection standards directly handle the physical protection/encoding of content. Such standards may be of a low-level type, such as cryptographic algorithms and watermarking standards, of an intermediate level, such as the smart media protection standards, and of a high-level type such as MPEG-4 IPMP (which in turn employ low level features of the low level type).

Some relevant examples of content protection technologies are;

- the Content Scramble System ("CSS"), which is employed to protect the media content DVD video disks;
- the High-bandwidth Digital Content Protection (HDCP) whose goal is to protect high definition content during transmission from a source device to a display device;
- the Digital Transmission Content Protection, (DTCP) whose objective is to restrict the capabilities of "home based" digital technologies, such as DVD players and televisions, through the encryption of interconnections between devices.

### *3.2.2 Basic Topology*

DRM systems may present multiple technical variations between them. Their structures may differ, the base technologies they use may vary, the employed content encoding schemes may differ and these systems may have overall different aims. Still there are a number of basic characteristics which are generally common to all DRM systems, and thus these comprise a relatively standard set of necessary logical components.

DRM systems may be analysed according to two different perspectives [85]:

- an architectural and entity bound perspective;
- a functional and procedural perspective;

From the architectural and entity bound perspective, a generic DRM system will comprise the following entities and components [89] (as presented in Figure 11):

- Content Owner – The content owner is the entity which inserts the consumable content into the system. It provides the digital goods to the packaging entity, and the contracts and rights related information to the License Server. This latter data will determine the type of content usages which are allowed;

- Packager – The packager handles the packaging and encrypting of the original content. It takes care of the following tasks:
    - Data compression – certain types of digital goods dispense with compression, and thus this task is optional;
    - Content protection – this includes encryption, watermarking, etc.
    - Key Supplying – supplies the content decryption keys to the Key Distribution Server;
    - Protected Content Supplying – supplies the protected goods to the Content Distribution Server/System;

- Key Distribution Server – The Key Distribution Server distributes the content decryption keys, (received from the Packager), to authorized users/clients. User/client authorization for key reception is obtained from the License Server;

- Content Distribution Server/Systems – The Content Distribution System/Server makes available the actual consumable digital goods as well as information about those goods (products or services). These systems may employ many different means to perform the distribution of goods;

- License Server – The License Server:
    - manages licensing information. It produces and delivers licenses expressing rights, and permissions, over digital goods for users/clients based on:
        - Information received from the AAA (Authorization, Authentication and Access control) Server and/or Payment;
        - Contracts and rights related data received from the Content Owner.
    - supplies user/client authorization information to the Key Distribution Server;

- AAA (Authorization, Authentication and Access Control) Server and/or Payment – The AAA Server handles authorization, authentication and access control aspects. Upon user purchase of rights it informs the License Server to provide the user/client with the respective license;

- Usage Tracking Server – This server gathers content usage information received from the License Server, and possibly the client side, and supplies it to the Usage Clearinghouse;

- Usage Clearinghouse – The Usage Clearinghouse surveys content usage and handles the processing and distribution of that data to all involved parties e.g. Content Owner, Financial Clearing House;

- Financial Clearinghouse – The Financial Clearinghouse enables financial transactions to be performed. It collects the revenues and performs the distribution to all involved parties, e.g. Content Owner, etc.;

- Users, Devices and Clients – Content usage rights are more appropriately associated with users, than with devices. This is so because more than one device may be associated with a given user, and also because devices may be connected in either a permanent or intermittent way.

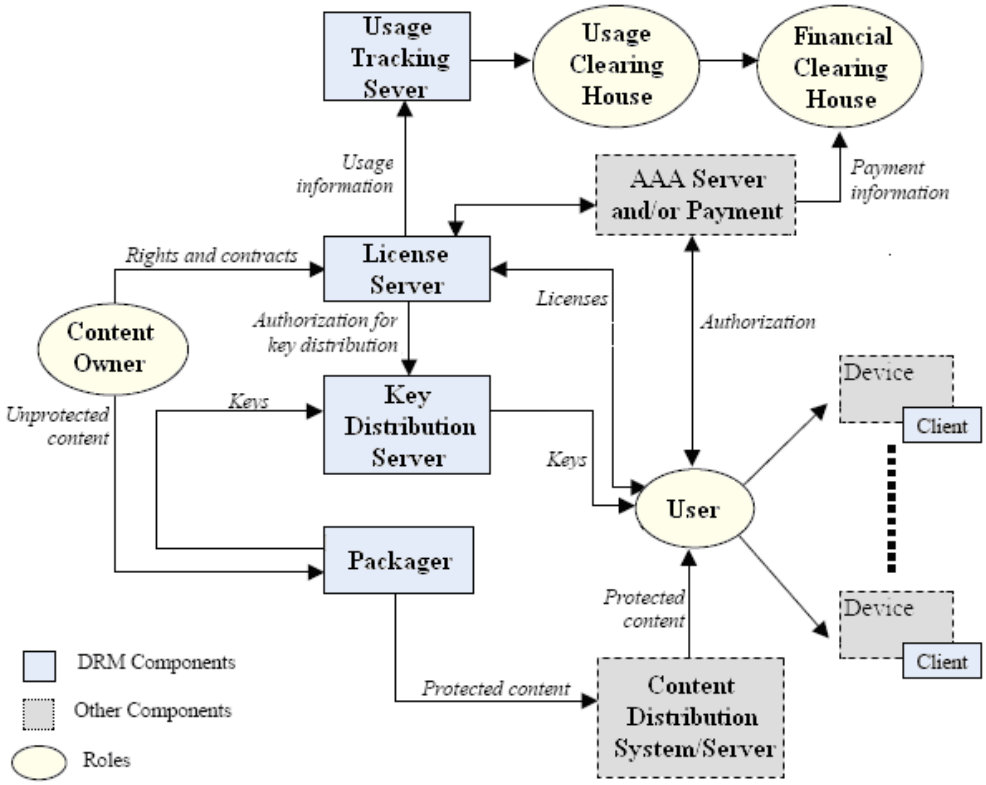


Figure 11 – Basic DRM Architecture (adapted from [89])

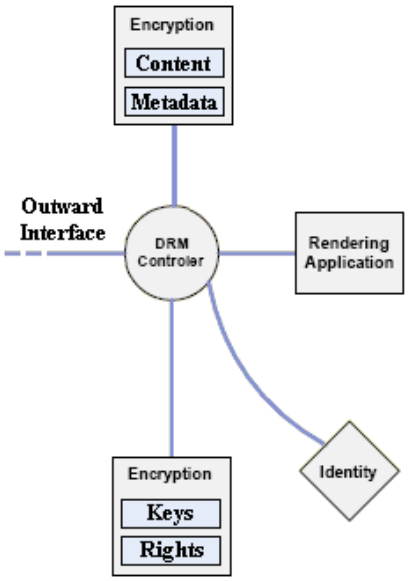


Figure 12 – Basic Client Structure (adapted from [85])

The client side of the system should supply the following functionalities [85].

- DRM controller – assures the enforcement of both user and Content Owner rights over content;
- Content, metadata, cryptographic;
- Content rendering;
- User authentication.

Figure 12 presents a graphical overview of the generic client architecture.

The steps involved in the accessing of content typically include [89]:

- The reception of the content in a protected form by the user/client, resorting to any of the methods of delivery supported by the content distribution system;
- The accessing of the AAA server, by the user, to obtain authorization, authentication and access permission and/or to proceed to fee payment. The subsequent signalling, by the AAA server, of the License Server, so that the latter delivers the license to the user, and also authorizes the Key Distribution Server to deliver keys;
- The delivery of keys by the Key Distribution Server so that the user (client side) may begin content consumption;
- The decryption of content, its consumption and the related usage control at the client application.

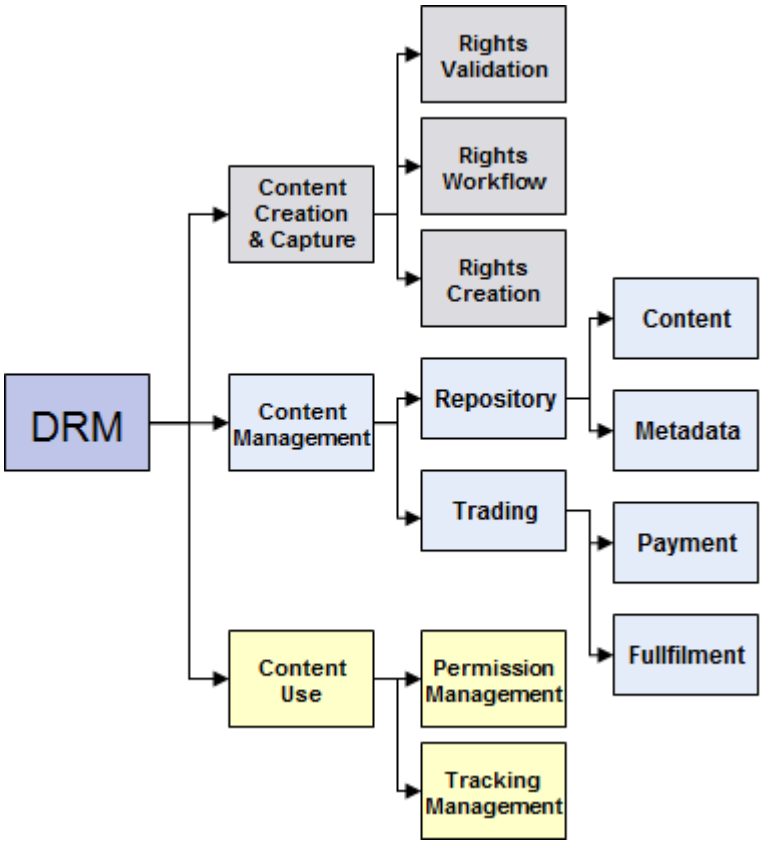


Figure 13 – Typical DRM Functional Architecture (adapted from [85])

From a functional perspective, the key functions of any DRM architecture are (as presented in Figure 13) [85]:

- Content Creation and Capture: Managing the creation of content to optimize trading. This function of DRM systems includes provisions for:
  - Rights validation – to ensure that content developed from existing content includes is being generated in accordance with the appropriate rights to do so and that such rights are consistent;
  - Rights creation – to assign rights to new content;
  - Rights workflow – to process content for review and/or approval of rights.

- Content Management – Managing and enabling the trade of content. This functional aspect of DRM systems should include provisions for:

  - Repository functions – to assure the access to content and its descriptive and rights related “metadata”;
  - Trading functions – assigning licenses to actors who fulfil the necessary conditions to have assigned onto them specific rights over content.

- Content Usage – Managing the usage of content after access to it has been granted. This function of DRM systems includes provisions for:
  - Permissions management – enforcement of rights associated with the content;
  - Tracking management – content usage monitoring where such tracking is a requirement of the user’s agreement.

## 3.3 Specifications and Implementations

### *3.3.1 Specification Initiatives*

#### *3.3.1.1 OMA DRM*

The OMA DRM specification was created by the Open Mobile Alliance [103]. This DRM scheme, meant for use with mobile-centric content types, was developed to allow mobile content providers to add DRM to their products.

OMA DRM 1.0 consisted of a basic DRM standard without strong protection measures. It manly defined three key methods:

- Forward Lock – focuses on the delivery of digital goods that must not be forwarded. It frequently applies to subscription-based services, were the involved devices are allowed to play, display or execute, but cannot forward the media object [104];

- Combined Delivery (combined rights/media objects in the DRM message) – enables the setting of specific usage rights over the digital good. This method extends the Forward-lock method by allowing a more refined definition of rights, which can be restricted by using both time and count constraints [104];

- Separate Delivery (separated Rights Object (RO) + encrypted media object in the DRM message) – provides the necessary provision to protect higher value media and enable the superdistribution of digital goods, by permitting the client devices to forward the media, but not the associated rights. According to this method the media and rights are distributed via separate channels. The media is encrypted employing a symmetric cipher, while the rights hold the encryption/decryption key [104].

OMA DRM 2.0 is an extension of the 1.0 separate delivery mechanism. Each device taking part in an OMA DRM 2.0 system possesses an individual DRM PKI certificate with a public key, and the corresponding private key. Rights Objects, which carry the content decryption keys, are individually protected for a specific receiving device by encrypting them with the device’s public key.

For a device to be eligible to receive a RO, it must first register with a Rights Issuer entity so that its certificate is validated. Devices known to be hacked can be impeded from accessing content.

The overall operation of an OMA DRM compliant system is as follows [105]:

- The content is packaged and protected from unauthorised access, by the Content Issuer, and the corresponding RO is produced by the Rights Issuer. Content and rights data is then inserted into the system;
- The content is delivered to the terminal side under any means, but the rights data is distributed by the rights issuer in a safe and controlled fashion;
- At the consumption device a trusted DRM Agent is installed which enforces the rights objects, which cryptographically are bound to it.

The basic steps undertaken by the OMA DRM architecture to assure the secure distribution of content preventing unauthorized access are [105]:

- Content packaging – content is packaged in a secure content container, and encrypted with a symmetric content encryption key (CEK);
- DRM Agent authentication – DRM Agents have a unique private/public key pair and a certificate which allow their secure identification;
- Rights Object generation – Rights Object are XML documents which express the permissions and constraints associated with the content's usage. They carry the CEKs;
- Rights Object protection – sensitive parts of the Rights Objects are encrypted before delivery (the CEK for instance), and the object itself is cryptographically linked to the target DRM Agent. Furthermore the Rights Issuer digitally signs the RO;
- Delivery – the RO and protected content package, being inherently secure, can be delivered using any transport mechanism, either together or separately.

OMA DRM employs its own rights expression language (OMA REL) for rights expression.

This DRM specification has been implemented on numerous cell phones. Furthermore, multiple mobile operators (such as Vodafone [106], Vivo [107], etc) use OMA DRM for their content delivery services. Commercial suppliers of OMA DRM solutions include, for instance, Beep Science, CoreMedia DRM, Discretix, Mutable OMA DRM, SafeNet and Viaccess.

#### *3.3.1.2 OpenSDRM*

OpenSDRM specification defines an adaptable DRM system which may be employed for various different business models and for the protection of different types of content. The adopted security architecture is based on the OPIMA [108] international specifications, on MPEG-4 IPMP Extensions, on MPEG-21 IPMP architecture and on some of the proposals for JPEG2000 standard Part 8 [85].

OpenSDRM was initially developed within the scope of the FP5 EC project MOSES. This project focused explicitly on the MPEG-4 file format for the protected content. Still this system was conceived so that it can handle all types of content and various business models (both for download, streaming or even broadcasting).

Figure 14 presents a graphical overview of the OpenSDRM platform architecture.

The components and actors that interact externally with the Open SDRM architecture are [85]:

- User – human individual who wishes to consume some content. The user will interact with Open SDRM in order to, identify himself, acquire licenses and play multimedia;

- IPMP Tools Provider – organization that creates tools for encryption, scrambling, watermarking, etc, for content protection and makes them available to the Open SDRM system. This supplying of tools implies the existence of business relationships between Content Provider and specific IPMP Tools Providers, since that producers

and/or distributors of content will specify which type of protection the content will have and which tools can be applied to the content and from which supplier;

- Content Provider – multimedia assets supplier that feeds Open SDRM with content and/or metadata;

- Payment Infrastructure – structure which facilitates Open SDRM e-commerce features through the provision of services for handling electronic payments;

- Certification Authority – entity which is responsible for the reception of requests for and issuing credentials to, other entities in the system. These credentials will be utilized by system entities to authenticate themselves before each other, enabling the establishment of secure and authenticated communication channels between them. The components in the Open SDRM architecture communicate with each other through the channel security provided by the SSL/TLS protocol. This Certification Authority can also be internal to Open SDRM, and thus entirely managed by some entity, or it may be an external commercial Certification.

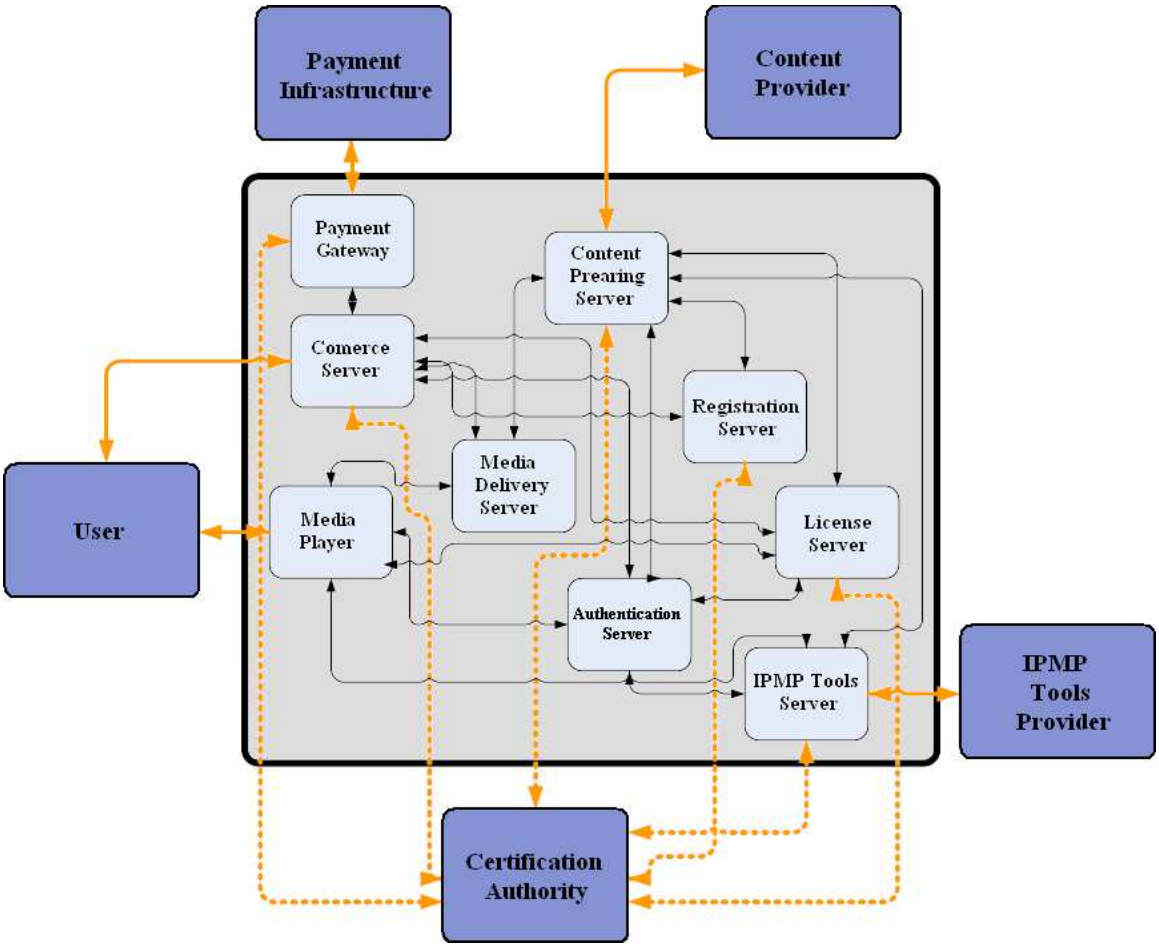


Figure 14 – OpenSDRM Solution Architecture (adapted from [85])

The internal components of the Open SDRM system include [85]:

- Content Preparation server – this server side component handles content preparation. It receives raw content from a specified source or sources and encodes it on a specified format, adds metadata and protects it;

- Commerce server – this component handles the trading of the content with the users;

- Media Delivery server – this module handles the delivery of digital assets to the client. This Media Delivery server will implement a specific protocol (download (FTP, HTTP, or other), streaming (RTSP, other), broadcast) to exchange protected content with the client application;

- Registration server – the role of this component within the system is to assign unique identifiers to content and to register metadata information for that specific content. The content identification scheme employed follows the MPEG-21 directives about Digital Item Identification (DII), using a reduced version of the MPEG-21 DII Digital Object Identifiers (DOI);

- Authentication server – this module handles the authentication of all internal and external entities to the DRM system. It's role is to validate the access rights of all the entities and components in the system working as a SSO point, registering and managing components and users on the system. It employs cryptographic XML credentials for the authentication of both components and users in order to authenticate the transactions exchanged between them (XML Encryption and XML Signatures);

- License server – this component is responsible for house-keeping the rules associating a user, the content and his/her corresponding access rights. It accepts connections from authenticated client Media Players for downloading of licenses, which will be applied to the protected content through an appropriate IPMP tool. The licenses are XML formatted and employ Open Digital Rights Language, (and, in the future, possibly the Rights Expression Language by MPEG-21;

- IPMP tools server – this server component handles the registration of new IPMP tools and the reception of the authenticated client Media Player requests for the downloading of specific IPMP tools. It is also a part of its role to make IPMP tools available to the Content Preparation Server to allow the protection of content;

- Media Application – this represents the applications that will be utilized to consume the digital assets. This generic component is able to display/playback the appropriate content. It may work with one or several IPMP tools in order to control how the content is accessed by a particular user.

#### *3.3.1.3 ISMA/DRM*

The Internet Streaming Media Alliance [109] is composed by companies from multiple areas. Its founding members are Apple Computer, Cisco, IBM, Kasenna, Philips and Sun Microsystems Inc [89].
ISMA's protocols are based on open specifications for media formats and transport of media content to the receiver over IP networks. This specification conforms to the ISO/IEC 14496 (MPEG-4) standards over Real Time Transport Protocol (RTP) [89].

ISMA has undertaken work to add DRM support to its provisions. ISMA/DRM preserves ISMA's interoperability goals, employing standard encryption, authentication and integrity validation for ISMA conforming media and protocols.

Figure 15 presents a succinct view of the interactions flow within the ISMA DRM architecture.

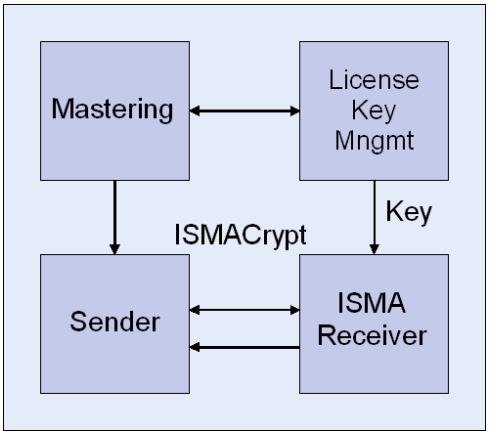


Figure 15 – ISMA DRM Architecture (adapted from [110])

ISMA DRM's components and their roles are [110]:

- The Mastering entity handles content preparation and publication;
- The Key/License Management entity manages and delivers the licensing and decryption data;
- The Sender entity handles the delivery of the content to the ISMA Receiver via an open-standard protocol called ISMACryp;
- The ISMA Receiver processes the ISMACryp encrypted streams, authenticated messages, and signalling.

More recently ISMA has merged with MPEG Industry Forum (MPEGIF), under the name MPEGIF. Its assets (and thus ISMA DRM), passed to the control of MPEGIF [111].

#### *3.3.1.4 MPEG IPMP*

##### *3.3.1.4.1 MPEG IPMP Extensions*

Intellectual Property Management and Protection Extension (IPMP-X) is a DRM architecture developed by the MPEG working group. This architecture provides a normative framework to support requirements such as: renewability, secure communications, verification of trust, granular and flexible governance at well-defined points in the processing chain, etc.

Two types of IPMP Extensions exist [85]:

- MPEG-2 IPMP-X which is meant to be applied to MPEG-2 based systems;
- MPEG-4 IPMP-X which is intended to be applied to MPEG-4 based systems.

MPEG-4 IPMP Extension specifies five key elements [112]:

- IPMP Tools – modules which handle (one or more) IPMP tasks such as authentication, decryption, watermarking, etc. A specific IPMP Tool may coordinate other IPMP Tools. Each such tool possesses a unique IPMP Tool ID that identifies it an unambiguous way, either at a presentation level or at a universal level;

- IPMP Descriptors – subsection of the MPEG-4 object descriptors (OD) which describe how to access and decode an object. These Descriptors are used to describe the IPMP Tool that is used to protect the object. An independent registration authority (RA) is utilised so that any party may register its own IPMP Tool and identify this without conflicts;

- IPMP Elementary Stream (ES) – transports IPMP specific data such as key and rights data. All MPEG objects are represented by elementary streams, which can

reference each other. These special elementary streams may be employed to deliver IPMP specific data;

- IPMP Tool List – this list carries information pertaining to the tools required by the terminal so that it can consume the content. It is conveyed in the Initial Object Descriptor (IOD) of the MPEG-4 system stream. Using this mechanism the terminal is able to select and manage the tools, or to retrieve them when they are missing;

- Secure Messaging Framework – this framework does not specify functional interfaces. It is instead based on secure message communication and mutual authentication. Interaction between the terminal side and the IPMP Tools is performed through messages via a conceptual entity called “Message Router”.

#### *3.3.1.4.2 MPEG-21 IPMP*

A key goal of the MPEG-21, multimedia framework is to enable transparent and augmented use of multimedia resources across a wide range of networks and devices. MPEG-21 Part-4 defines an interoperable framework for Intellectual Property Management and Protection (IPMP).
This framework includes standardized provisions for:

- retrieving IPMP tools from remote locations;
- exchanging messages between IPMP tools and between these tools and the terminal;
- authentication of IPMP tools;
- integrating Rights Expressions according to the Rights Data Dictionary and the Rights Expression Language [113].

It does not, however, define protection measures, keys, key management, trust management, encryption algorithms, certification infrastructures or other components.
This part of MPEG-21 defines how to include IPMP information and protected parts of Digital Items into a DIDL (MPEG-21 Digital Item Declaration) document. The IPMP DIDL encapsulates and protects a portion of the hierarchy of a Digital Item, and associates appropriate identification and protection information with it.

MPEG-21 IPMP consists of two parts:

- The IPMP Digital Item Declaration Language – provides the tools for a protected Representation of the DID model. It permits the expressing of an encrypted DID hierarchy which is digitally signed or otherwise governed to be included in a DID document in a schematically correct manner;
- IPMP Information schemas – defines structures for the specification of information pertaining to the protection of content, including tools, mechanisms and licenses.

#### *3.3.1.5 DMP DRM*

The Digital Media Project (DMP) was established with the purpose of promoting the deployment and use of Digital Media in ways that respect the rights of creators and rights holders to exploit their works [114]. In that regard it has performed the specification of a standard for “Interoperable DRM”.

DMP DRM attends to the issue of DRM Interoperability through the specification of individual technologies (called Tools within DMP context) necessary for the implementation of, DMP termed, “Primitive Functions”. Such tools are to be “smaller” functions obtained by breaking down into atomic procedures the actions performed by value-chain users when they do interact between themselves. While more general functions may undergo substantial

changes as a consequence of the evolution of the media business in the value-chain, these more “Primitive Functions” will remain more stable and be more easily augmented.

Therefore, the DMP DRM is not meant as a universal “DRM standard”. This project specifies tools for enabling Primitive Functions [115]. Its development is a progressive process which is opened to exterior contribution. In that regard, calls for proposals for additional necessary tools to support new primitive functions or additional functionalities of existing tools are periodically issued.

### *3.3.2 Implementation Initiatives*

#### *3.3.2.1 Windows Media DRM*

Microsoft’s Windows Media DRM (WMDRM), is a Digital Rights Management service for the Windows Media platform. It is based on a Microsoft proprietary specification which was designed to provide safe delivery of audio and/or video content over an IP network to PCs or other devices, enabling distributors to control content usage.

##### *3.3.2.1.1 WMDRM Operational Overview*

Figure 16 presents an overview of the architecture of the Windows Media DRM system (more specifically the Windows Media Rights Manager part of WMDRM). It also highlights the main differences of that architecture when compared to the typical one presented in chapter 3.2.2.

Digital goods protected by the WMRM system, are distributed, over the Internet, in a protected and encrypted file format.

The characteristic WMDRM action flow is as follows [116]:

- Packaging – The media content files are encrypted with a key. That key is stored in an encrypted license, which is distributed separately. The media file also carries other information such as the location (URL), where its respective license can be purchased;

- Content Distribution – The packaged files may be distributed by whatever means available, because their protection scheme assures that (in principle) no illegitimate access take place;

- Establishing a License Server – A License Clearing House (LCH) is selected by the content provider (or content owner). The LCH will store the specific rights or rules associated with the licenses and implement the WMDRM license services. The LCH is responsible for the authentication of consumer's requests for licenses. In Figure 16 the LCH is represented as the association between License Server and Key Distribution Server;

- License Acquisition – For the consumption of a WMRM protected digital good to take place, the consumer must first obtain the appropriate license key to unlock the file and access its media content. For the license acquisition process to take place the WMDRM system sends the consumer to a registration page (AAA Server) where information is requested or payment is required, or "silently" retrieves a license from a clearing house;

- Playing the Media File – WMDRM protected files must be rendered by a WMDRM supporting media player. Employing such tools, the consumer can access the media

content according to the specific rules or rights that are included in the respective license. Licenses are not transferable. They are valid for a single PC, or device.

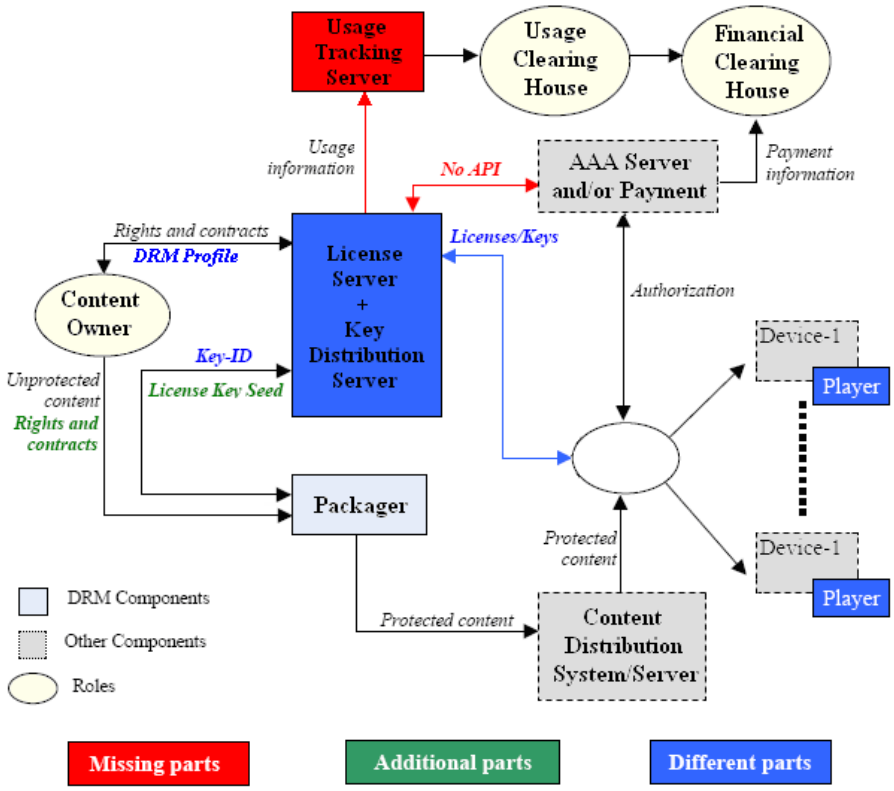


Figure 16 - Windows Media Rights Manager Architecture (adapted from [89])

Content usage rights are associated with devices instead of users. Still, moving content to a mobile device is possible, but there is no general support for multiple devices.

#### *3.3.2.1.2 WMDRM Content Protection Scheme*

The content producer (and packager) packages the media content into a key protected (encrypted) file. The encoding is performed using Windows Media Encoder [85]. In order to be able to use Windows Media Encoder, a content owner must possess a DRM profile obtained at a LCH. That profile allows a content owner to generate keys for content encryption as well as to define the terms for the protected content's usage.

During the content protection process, a key is generated. That key is employed in the content ciphering operations. The key is produced using a specific algorithm which employs a license key seed (which is conveyed to the content owner and packager by the mentioned DRM profile) and a key ID. This way, the license key seed is shared between the content owner and the LCH. The key ID is generated by the content owner and it is included in the header of the protected file/stream.

Furthermore the content header also carries other information such as [85]:

- The URL of the LCH;
- A unique content ID which identifies the content, and content related information (for instance artist name, date of recording, etc.);
- The version of the individualized DRM component;
- Attributes of type "name-value" defined by the content owner to carry other necessary information.

Additionally, the header, of the protected content, is digitally signed, using a public/private key par, in order to ensure the legitimacy of the header and its information.

To issue licenses for specific contents, the LCH, regenerates the appropriate key by retrieving the key ID from the packaged file and the key seed and performing the previously mentioned algorithm. The key is then included in the license file sent to the consumer's computer.

Using the key enclosed in the license, the WMDRM compliant player on the consumer's PC may then access the protected digital goods.
When protected content consumption is attempted, via Windows Media Player for instance, this application will search the local PC for a valid license. If the latter isn't available, the player analyses the content header in order to retrieve the URL of the responsible LCH and the key ID. Using these two values the client application obtains a valid license containing the decryption key.
Still, if the content is to be paid, the end-user will first be directed to a virtual location where he must proceed to purchase the license [85].

Once a valid license is available, the terminal device rendering software can them render the content, but in accordance to the rights and permissions defined in the license.

#### *3.3.2.1.3 WMRM Licenses*

Licenses contain the keys to unlock the Windows Media files. Furthermore the licenses are the vehicles which convey the rights, or rules information related the governing of content usage, from the content owner side to the consumer side. The WMDRM system can support licenses describing a wide variety of different usage rules, such as:

- the number of times a digital good can be consumed;
- the devices which can render or process a content;
- the time at which the user can start accessing the file and it expiration date;
- if the file can be transferred to a CD recorder;
- if the user can back up and restore the license;
- the security level required to have on the client to access the Windows Media file;
- etc.

### *3.3.2.2 Helix DRM*

Helix DRM (formerly RealSystem Media Commerce Suite) is a DRM system developed by RealNetworks. This system adds DRM functionality to a number of other RealNetworks media components, such as Real player and Real streamer.

Figure 17 presents an overview of this system's architecture. It also highlights the main differences of that architecture when compared to the typical architecture of a DRM system as presented in chapter 3.2.2.

This DRM system consists of the following main components [89]:

- Packager – handles the content packaging and encryption measures. It receives as input data the unprotected media files, the public key of the License Server at stake and the URL for the Retail Server. It produces the encryption keys, unique content IDs and delivers the packaged and secured media files. The secured files include, besides the encrypted media, the relevant media related metadata such as the retailer URL and content ID generated by the packager [89]. No usage rules are included into the protected media file by the packager. The public key of the License Server is used for the secure transmission of the content encryption key [89];

- Client application – is composed by the Real player with a DRM plug-in. It handles the usage tracking and usage clearing procedures. When the consumption of a protected media asset is desired, the Real player searches for a local license for the content at stake. If it is not available, the player invokes a web browser and calls the retailer URL (which is included in the protected file as metadata) [89]. If all license acquisition procedures go well, the retailer provides a valid license to the client side. Such a license contains usage rules, which will be interpreted and enforced by the plug-in;

- License Server – offers an http interface and a set of procedures for the creation of licenses. It receives as input the content ID, the content encryption key, the usage rules, and the client's public key (which enforces the personalization of the client). Employing those parameters, the License Server generates a license containing the usage rules and the encryption key, encrypted by the client's public key [89]. The License Server does not interact directly with the consumer. The Retail Server will be the intermediary between the two.

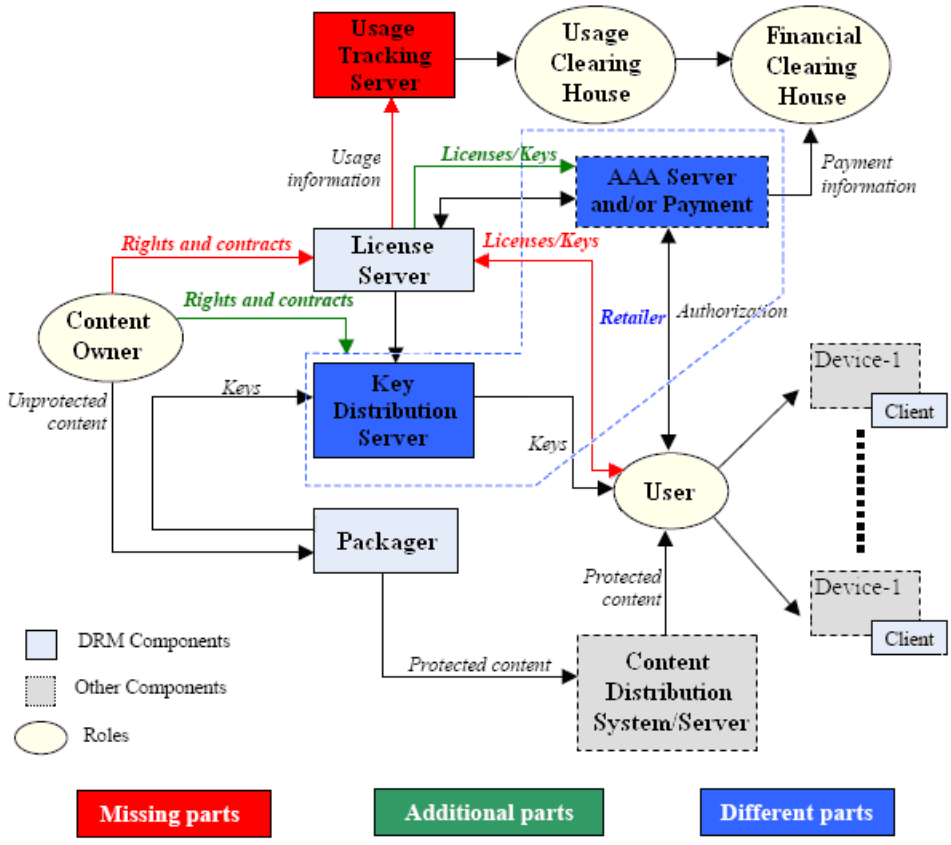


Figure 17 - RealSystem Media Commerce Suite Architecture (adapted from [89])

The full functionality of the Helix DRM system requires the existence of a Retail Server. Such a component functions as the gateway to the payment system and to the license server, completing the business workflow. Still it is not included in the system. Nonetheless, it is to link with the interfaces of the packager and the license server [89].

The Retail server should operate in close association to a database which stores content IDs and their respective content encryption keys[89].

A typical behaviour for the Retail Server is for it to call a payment system (under the request of the user side), and, after a successful financial transaction, to proceed to license creation.
For license creation to take place, the Retail Server is responsible for the definition of the usage rules, and for forwarding them, together with the content ID, the content encryption key, and the client's public key (delivered at content purchase time by the client side DRM plug-in) to the license server [89].
After a valid and client specific license is created it is forwarded to the client plug-in so that content consumption can take place.

### *3.3.2.3 DMDFusion*

DMDFusion is an open, flexible and scalable DRM solution for carrier grade servers. It is a complete DRM solution as it includes rights creation, management, delivery and enforcement of usage rules [85]. It also integrates multiple proprietary DRM solutions such as those from Microsoft, Adobe and Real Networks [89], which can interact with it through a language-independent API.

The DMDFusion system is entirely server based, and thus, vastly different types of clients, such as set-top boxes, mobile devices, etc., can consume DMDFusion compatible content without the need for any 3rd party DRM solutions.

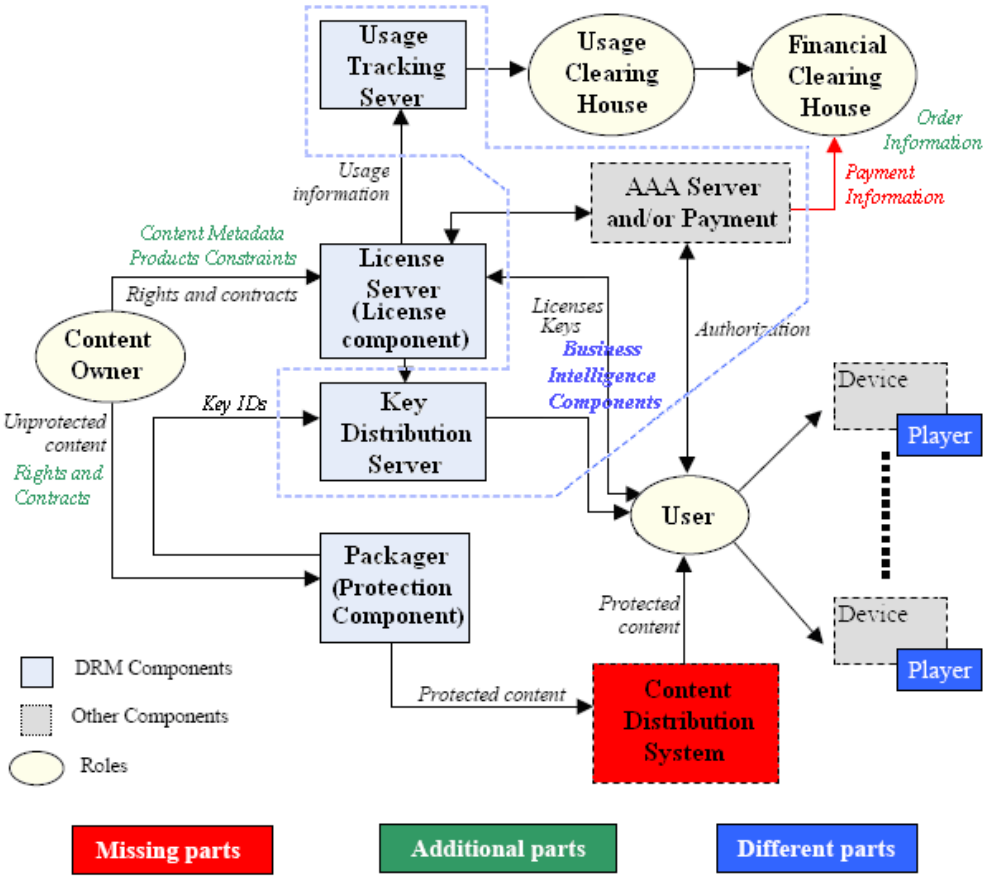


Figure 18 – DMDFusion Architecture (adapted from [89])

The DMDFusion architecture is composed by three main components as presented in Figure 18 [89]:

- The protection component (fulfils the role of a Packager) – protects content, employing specified technology and delivers the protected assets to a client specified location. It permits both push and pull protection models [89], that is, the request for

content protection may originate from the content owner or from the requester of the content, (for instance the end-user);

- The license component (fulfils the role of a License Server) – handles the creation and issuing of licenses, for a specified technology. For instance, a request from a Windows Media player will provoke the issuing of a Windows Media license;

- The business intelligence component (or rights management component) – stores and manages the business information that the protection and license components need for their operations. It is responsible for handling all the aspects related to rights management, and it provides a web based GUI, that enables content owners to define usage rights, server side conditions and contracts.

Keys for content owners are DRM-specific. They are generated by the protection component employing the appropriate DRM technology. These keys are then stored in the business intelligence component.
The key pairs (public and private) for the Content Distributors, on the other hand, are generated by the Content Distributors themselves. The public keys are stored in the business intelligence component.
Given that the business intelligence component handles the storing and management of keys, it thus fulfils also the role of a Key Distribution Server [89].

The requirements for terminal devices depend on the type of DRM technology being employed for content protection. Generally, PCs, PDAs, etc., equipped with the appropriate player/reader software running the appropriate DRM components may be used. The set of players (and respective DRM schemes) for which this system has support includes Adobe Reader, Windows Media Player and Real Networks Player [89].

In what concerns Authorization, Authentication and Access Control (AAA), besides enforcing rights using licenses, the DMDfusion system also possesses the capacity to restrict the issuing of licenses. For instance, license issuing may be restricted based on the geographical location of the license requester which is determined by checking its IP address. License issuing can also be restricted to a maximum number per product and/or to various time intervals. The verifications involved in the imposing of the previous restrictions are performed by the license component, which queries the business intelligence component for details about the required good, at the time of a license request [89].

Within the DMDfusion system, the only usage tracking that takes place is the recording of the delivery of licenses for particular contents. Reports can be provided on the issuing of licenses for specific contents, time intervals, geographical territories, etc. The usage that a content is subjected to once the user has acquired a license is not tracked by DMDfusion [89].

DMDfusion does not include a payment system of its own (something like a Financial Clearing House). Nonetheless a content distributor may request information from the business intelligence component on orders (licenses requested and issued) and use this data to produce invoices or request payments [89].

#### *3.3.2.4 Secure Digital Container DRM*

Secure Digital Container (SDC) DRM is provided by SDC AG [117]. This technology is approved by all key Recording Labels as Mobile DRM Technology for full length music tracks [85]. The SDC architecture [118] is composed by a server (packager) software and a client software.

This solution employs Java DRM technology and is based on a mobile code architecture. Because of this, the system is capable of packaging content together with code in a "container". This object works as a transport unit for content, software and code. The client application is carried within the container and is interpreted by the Java Virtual Machine at the terminal side [85].

Some of the most relevant aspects of this system are the following [85]:

- No client installation is necessary – the system packages content together with code inside a single object. This way the container object can contain the client application required to decrypt and render the content, thus no client installation is required. The client device needs only a Java Virtual Machine;

- Fractioned content – the content is spliced into two separate containers (the specific characteristics of this fractioning make it advantageous for superdistribution):
    - Container A – Holds 95% of the content payload. The content can still be somewhat accessed, but the remaining space is filled with advertisement or promotion information. This container can be freely shared;
    - Container B – Container B holds the remaining portion of the content, (missing in container A), in an encrypted and watermarked form.

Figure 19 presents an overview of SDC DRM's architecture.

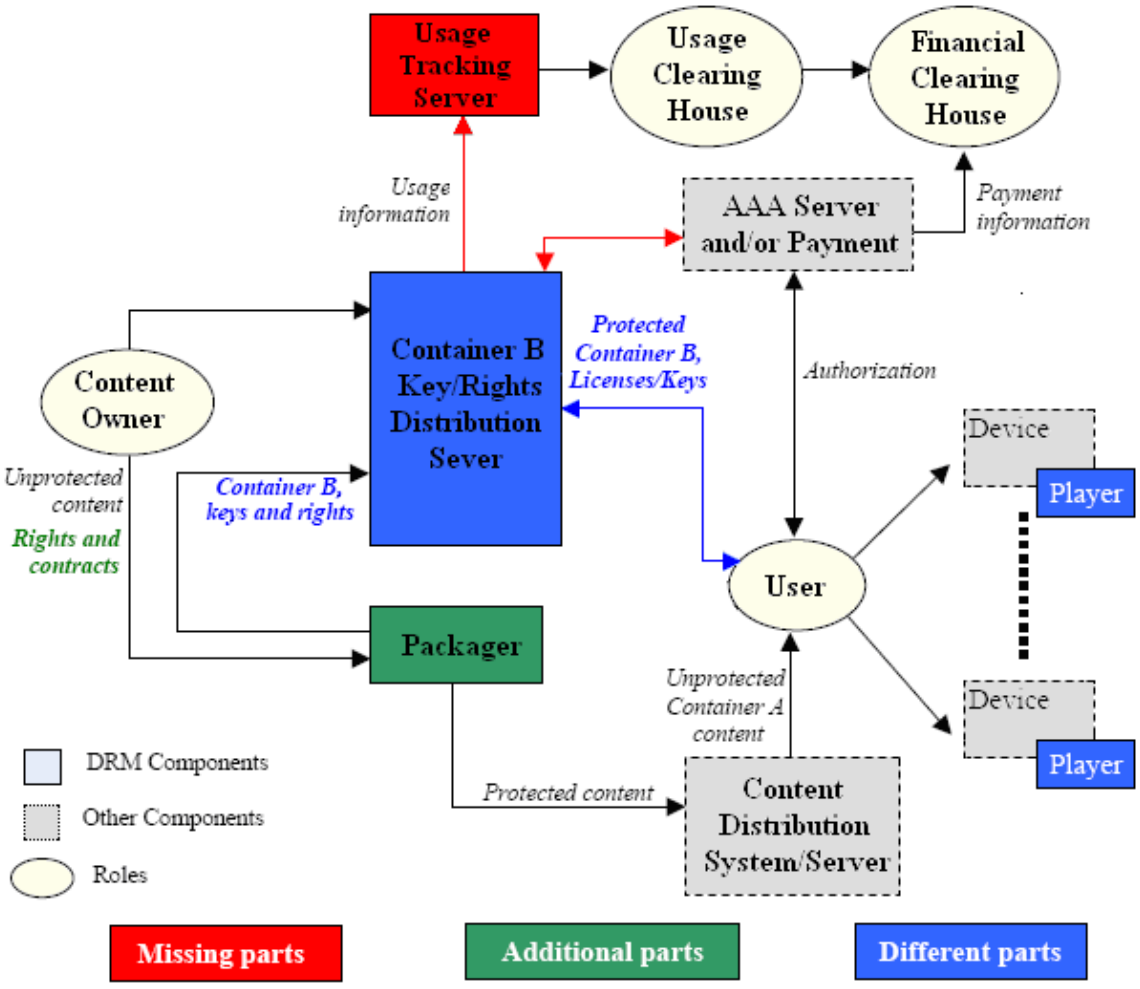


Figure 19 – SDC DRM Architecture

The SDC packager, (implemented at the server software application), handles the splicing of content into Container A and Container B. The content usage related rights information is inserted in Container B. The content splicing method may be configured for a given application [89].

In what regards the roles of a License Server and of a Key Distribution Server, this system's server side handles the distribution of Container B as well as keys and licenses to terminal side. The distribution of this data is secure, and a unique key is associated for each device [89].

On the terminal side, the usage rights are associated with devices instead of users. Within the operation mode of this system the client software is downloaded with the content. It performs the content decryption based on PKI, and it also handles the merging of Containers A and B using its interpretation module. During the interpretation process the content's fingerprint is matched against the device ID. This permits the tracking of unauthorized content copying [89].

More recently this DRM scheme has evolved and some changes where performed to it. An independent client now exists, the Secure Download Manager, which must be installed at the client machine and which mediates the interaction with the system's server provisions.

#### *3.3.2.5 OpenIPMP*

The OpenIPMP system [119] is an open source implementation of the OMA DRM specification. It employs only open standards, one of the most the relevant of which is the MPEG-4 format. OpenIPMP encompasses user management and identification, content encryption algorithms and distribution channel protection.

The main concepts of the OpenIPMP system are [119]:

- User Management – The cornerstone of the system's security structure are the User Management mechanisms. OpenIPMP issues each user a Digital Certificate (or Digital Id) at registering time. The Digital Id, is issued and signed by the OpenIPMP Certificate Authority. It is a standards-based electronic file that uniquely identifies the user within the system, specifying the group(s) to which the user belongs. The employment of this ID also permits the establishment of secure and confidential communications with the OpenIPMP server components;

- Rights Management – OpenIPMP enforces a robust usages permission and constraint model that permits content owners to specify a wide variety of rights and respective constraints. For the expression of such rights and restraints OpenIPMP supports Open Digital Rights Language and MPEG REL;

- Content Identification and Management – The OpenIPMP system provides measures for the unique identification of content. The system implements one of the identification schemes proposed by the MPEG-21 standard. The unique identification of content allows the system to provide protection and tracking mechanisms during the entire lifetime of a digital good;

- Content Protection – For content protection and overall security, this system employs several open cryptography standards such as [85]:
    - Public Key Infrastructure (PKI) – used to ensure encryption and digital signature for securing digital contents over networked environments. The security and extensibleness of the PKI framework allows for OpenIPMP installations to seamlessly cooperate;

- X.509 Certificates – used to implement licenses. Each digital certificate is signed by the OpenIPMP Certificate Authority service for authenticity, and thus ensures that each OpenIPMP transaction is cryptographically certified;
- Asymmetric Encryption (e.g. RSA) – employed to protect licences and other key sensitive data. In the case of RSA it supports configurable RSA key sizes;
- Symmetric encryption – employed to secure the digital assets. The two main algorithms used are AES and BlowFish;
- Secure Sockets Layer (SSL) – employed for critical transactions such as License Acquisition and Content Registration, thus ensuring that information remains authentic during transit;
- Secure Storage (RSA Security's PKCS #12) – employed to assure that all user-specific information (Private Key, User Certificate or Certificate Authority list) is safely stored.

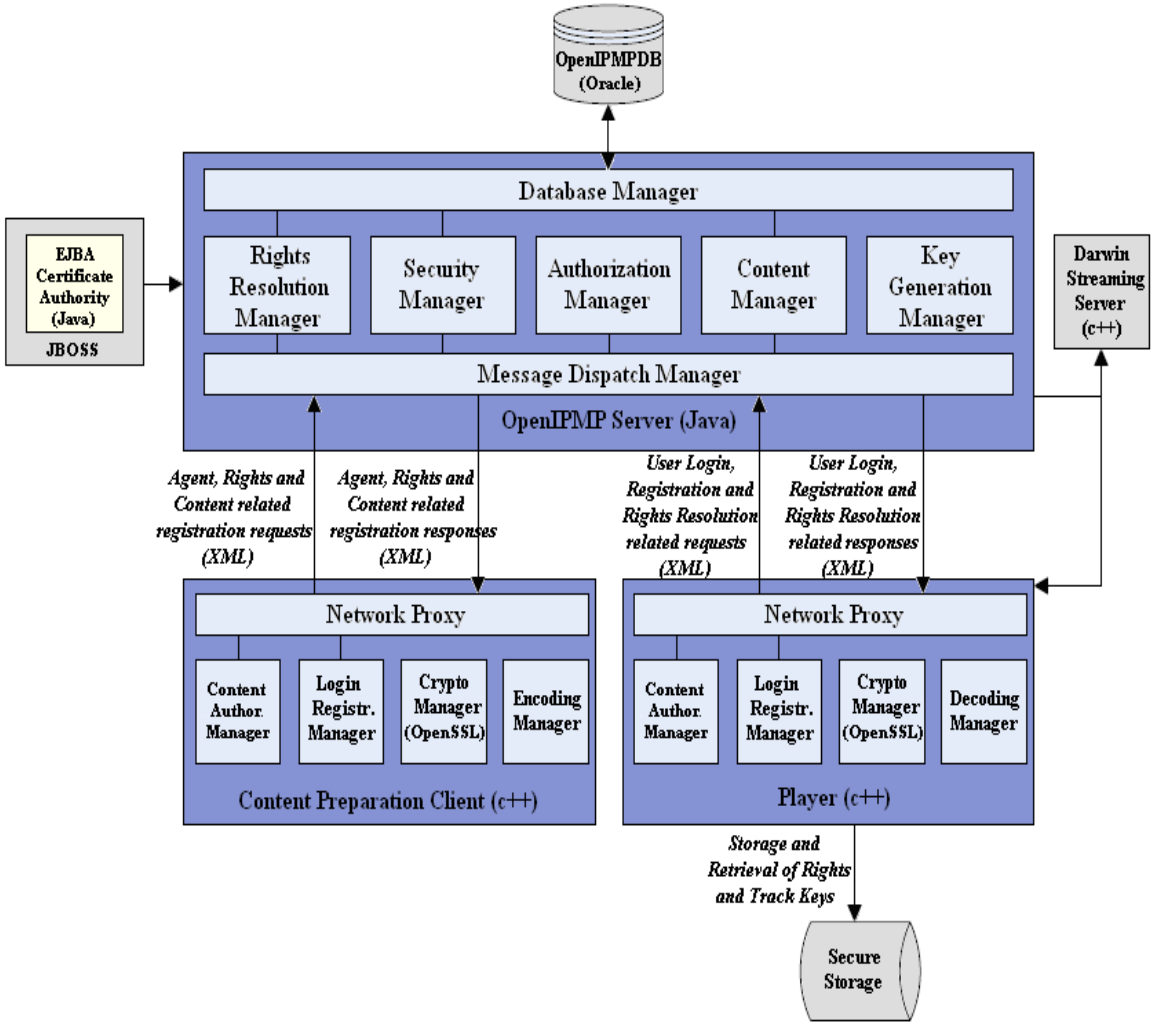


Figure 20 – OpenIPMP Components Diagram (adapted from [85])

The main components of the OpenIPMP platform are depicted in Figure 20. In operational terms, different sets of OpenIPMP's components combine or interact to constitute the following main tools or services [85]:

- Media Encoding tool – Employed for content protection. It handles all the necessary client side cryptographic algorithms to provide persistent protection of the digital goods;

- Media Player tool – This tool is used to playback OpenIPMP protected content. It deals with all the communication aspects with the server side components for license acquisition and also enforces usage rights;
- User Registration service – It is through this service that new users are registered in the system. When the registration process is successfully finished, the user is given its credentials which uniquely identify him and grant access to system resources;
- Content Management service – This service supplies registered users with the necessary tools to manage their contents. Upon completion of the content registration processes, a digital good is under the protection of the system. All communications between the OpenIPMP client side components and the Content Management Service take place through a safe communication channel;
- Rights Authorization service – Registered users resort to this service to define all the permissions for their registered contents;
- License Management service – This service handles all license requests from registered clients and deliver the respective authorizations;
- Administration tool – This tool allows an administrator to configure the system according to an organization's specific needs.

#### *3.3.2.6 AXMEDIS DRM*

The "Automating Production of Cross Media Content for Multi-channel Distribution" (AXMEDIS) [120] initiative is a research project, partially supported by the European Commission.

Besides other tools, AXMEDIS also delivers a multifaceted and interoperable DRM solution for employment with both B2B and B2C scenarios through traditional and P2P distribution structures.

This DRM solution focuses on the protection and management of rights for a vast set of different types of content, such as single files or complex cross media and multimedia, distributed on different channels to different kinds of terminal devices [121].

AXMEDIS DRM employs and extends the MPEG-21 framework in order to enable [121]:

- the protection of content of various formats and types;
- the controlling of the exploitation of rights of the above contents, using formal metadata licenses expressed in the MPEG-21 REL standard;
- the collecting and reporting of information about consumption of rights for accounting, billing and/or statistical analysis;

The DRM system delivers [121]:

- tools for content packaging and protection (simple manual tools, GRID technology based automated tools, AXMEDIS Content Processing);
- DRM servers for:
    - controlling the exploitation of rights of protected content;
    - data gathering on the exploitation of rights;
    - interacting with an intellectual property ontology to ease the production and verification of licenses.
- players for protected content rendering on PC, PDA, STB/PVR, and AXMEDIS Java based Mobile. AXMEDIS players can be customized and hosted in WEB pages;
- tools for manual and automated production of licenses.

For integration of AXMEDIS DRM premises into a system, that system's content usage licenses are required to be supplied to the AXMEDIS DRM Servers via a Web Service call. Alternatively the system may delegate the license production activities to the AXCP GRID.

AXMEDIS content packages (also called AXMEDIS Objects), contain a variety of simple and complex media objects (or references to them) containing various forms of metadata and dynamic scripts. These may be produced with the employment of AXMEDIS Editor tools for MPEG-21 and AXMEDIS authoring (SMIL, HTML, MPEG-4, or any other kinds of digital resources), DRM, licensing, protection, packaging, workflow, playing, etc [121].

## 3.4 Summary

The technical landscape in the field of DRM technology is a very diverse one. Several standards have been developed and various implementations have been completed. These solutions generally provide robust levels of security and their operation focuses on a restrictive governance of content usage.

In spite of their security capabilities, these systems have, nonetheless, been frequently broken and circumvented by users [122], and subsequently patched in an iterative cat-and-mouse game.

Developments, in this technological area, have begun some time ago, still, their reception, on the part of consumers, has been very negative, and, consequently, their employment by on-line content delivery initiatives has been problematic.

# 4 Commercial P2P Distribution Survey

## 4.1 Introduction

In spite of considerable initial hesitation, several initiatives have begun to appear in the field of commercial, on-line content distribution, which employ (or have employed) a P2P operation mode. In the following sub-sections we expose some of the most relevant such initiatives.

## 4.2 Veoh

Veoh [123] was an Internet Television service run by a California based company. This service employed P2P technology (among other means), for the diffusion of commercial and user generated content.

The Veoh system offered its users two possible options for the consumption of the available video content. Viewers could obtain the content directly from the Veoh.com site via streaming, or they could use the VeohTV application. This was a P2P based software application that enabled the downloading and viewing of content of potentially much bigger sizes. Content publishers could use their PC to upload videos for distribution.

In regards to security provision, Veoh distributed both DRM protected and unprotected content. That is, it employed a parallel DRM structure, to enable user and peer authentication, content integrity and authenticity validation, and access control on some of its content.

It searched for years for a successful BM, but ultimately, and under legal pressure from Universal Music Group [124], declared bankruptcy in 2010.

## 4.3 Babelgum

Babelgum [125] is a free Internet TV service. Originally it employed a proprietary P2P streaming technology, but has eventually dropped it in favour of a client-server operation.

The platform employs DRM protective measures which include the encryption of the exchanged content data streams, time-limited licenses and PPV [126].

Babelgum is a privately backed project which focuses on professionally produced content. It enforces some content access restrictions based on user geographical location. Its BM is advertisement based, employing advertising revenue-sharing. Content owners receive a portion of advertising revenues. If no advertisement is associated to the content this system guarantees producers a minimum of $5 for each 1000 unique views [127] .

## 4.4 JOOST

Joost is an Internet based system for the distribution of TV content (and other forms of video), developed by the founders of Skype and Kazaa.

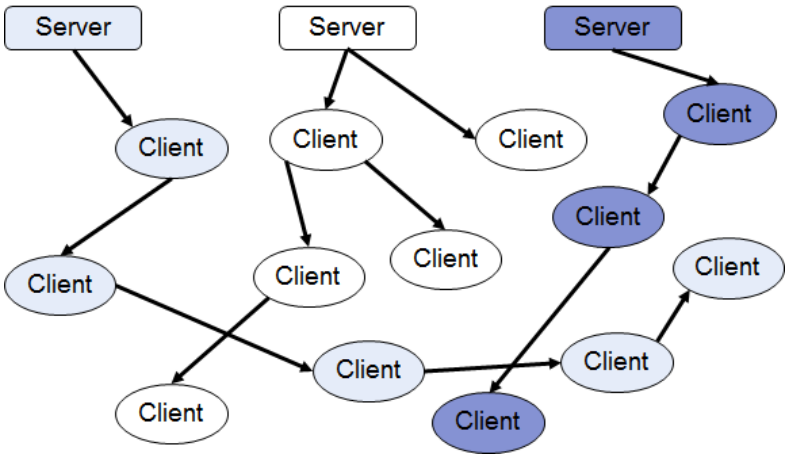


Figure 21 – P2PTV overlay network serving several video streams

It employs a P2P strategy (or did so initially), for the delivery of content, under proprietary DRM protection. The P2P technical layer was provided by the Joltid company, which also supplied the P2P infrastructure of Skype. Figure 21 presents a simplified image of Joost's P2P content diffusion network.

Content is distributed with the help of the nodes participating in the system, by taking advantage of their uplink capability, in a way similar to that of BitTorrent, and over a proprietary DRM connection, using a Joost specific encryption scheme.

Joost's employs an ad-supported BM, in a similar manner to that of regular TV, exposing users to both injected video-advertisements as well as additional interactive advertisements via overlays and short clickable pop-ups [128]. It was eventually faced with economic difficulties and remade itself as a "cost-effective" white-label video provider [129].

## 4.5 PPLive

PPLive [130] is a P2P network for the streaming of video content. It was develop in Huazhong University of Science and Technology in People's Republic of China.

The contents delivered by PPLive are generally targeted to Chinese mainland audiences. This service enforces no DRM measures and employs an add-supported business model.

## 4.6 ReelTime

ReelTime.com [131] was a video on demand provider that delivered movies and television shows over the Internet.

Content was delivered through a proprietary software system called Intelligent Rapid Delivery System. This employed some P2P networking to reduce the bandwidth demands on its servers. To this end, while the most part of the content files are delivered to the users by the system's servers, a fraction of such files are transferred between the system's terminal machines (i.e., peers). The delivered content was DRM protected.

ReelTime employed a subscription and pay-per-view BM. It has subsequently ceased to operate.

## 4.7 LiveStation

Livestation [132] is a platform for distributing live television and radio broadcasts, as well as on demand video, over the Internet.
It originally used P2P technology for said distribution, employing Microsoft Research technology [133]. The system divided a video stream into multiple stripes, each of which was shared independently among peers [134].
Livestation's business model is based on both advertising and fees [135].

## 4.8 Imeem

The original Imeem system begun as a distributed and peer-to-peer social network [136] for the sharing of files, photos and blogging data.

A client application was distributed as the primary platform, and the website was basically the means for users to obtain the client. Every client had its own database for the indexing of references to media content shared on the network. This contributed to facilitating the location of content. Users who wished to consume a specific content would access it through a direct exchange with the publishing peer.

Access to content was done on a permission base, and communication in the Imeem network was secured by the AES advanced encryption standard [136].
Its business model was predominantly ad-supported [137].

With time the service underwent many changes. Further features were added to the website and the client software ceased to be functional. The distributed database model was centralized, and all of Imeem's features became available without requiring a client download. The P2P architecture was annulled.

Finally, in 2009, after having been bought out by MySpace Music, it was shut down [138].

## 4.9 BBC iPLayer

iPlayer [139] is a service made available by BBC via Internet access (terminal application supported on Windows, Macs and Linux), cable television, iPhone, Nintendo Wii and iPod Touch [141], for the delivery of video and audio content.

The system originally employed P2P technology for the distribution of rich media content [142], but has since move away from it. When P2P technology was still employed, the network's peers where the iPlayer applications and the actual P2P infrastructure used was Kontiki [140].

For the protection of intellectual property rights, the system originally used Microsoft's Windows Media DRM [143] but has since moved to Adobe's DRM system. This technology

allows BBC to have a relatively fine control over the use of distributed content even after it has been transferred to the terminal iPlayer applications.

## 4.10 Qtrax

Qtrax [146] supplies a legal P2P music delivery service, built upon the Gnutella network [144]. It employs Microsoft's Janus DRM technology for content protection and access control and to enforce advertisement consumption.

It, originally, offered a two tiered service [145]:

- the first was a free, advertisement-supported tier designed to work with, and filter copyrighted content, from existing peer-to-peer networks;
- the second tier was a premium subscription based service which would require a monthly fee.

In time Qtrax has, however, moved to a strictly advertisement based BM.

Large music industry companies, such as EMI [145], have announced deals with Qtrax regarding music rights and publishing.

It has also faced legal and financial problems that have forced it to restart its operation, after a pause period [147].

## 4.11 Sky Anytime

Sky Anytime [148] is a package of services made available by BSkyB [149] for the delivery of video content.

One of the services, in that package, is a PC version of Sky Anytime which employs a broadband, Internet based, peer-to-peer structure (the Kontiki infrastructure), for content diffusion.

Content may be viewed for a certain number of days, and it is protected by digital rights management software provided by Microsoft [151].

The service is available at no extra cost to Sky subscribers and under a subscription fee for non-clients [150].

## 4.12 iMesh

iMesh [152] is a media content delivery system and an online social network. It employs a, Gnutella based, centralized P2P strategy for content distribution. For content protection the system uses Microsoft Digital Rights Management technology [154].

The system allows users to access large amounts of "non-copyrighted" content (video or audio) for free and makes copyrighted content available for a monthly fee in the form of either an iMesh Premium subscription or an iMesh ToGo (portable music) subscription. Users may also permanently purchase tracks [153].

## 4.13 TVUNetworks

The TVU Networks Corporation [155] operates an Internet based television broadcasting network which uses P2P technology.

Users download a freeware software application, called TVUPlayer, and may then join the P2P community and watch a varied number of channels.

TVU's solution includes monetization capabilities for content owners such as geo-filtering, subscription services and personalized in-stream ad insertion tools [156].

## 4.14 Zattoo

Zattoo [157] is an Internet based P2PIPTV content distributing system. It employs a proprietary peer-to-peer protocol. The terminal application (peer) runs on Mac OS X, Linux and Windows [158].

Zattoo current focuses on European channels. It delivers licensed content [157] under Digital Rights Management protection [158]. Zattoo ensures that [159]:

- Streams are protected by encryption;
- Streams cannot be copied as no copy of the stream is stored on the network;
- Streams cannot be retransmitted as sources would not be authenticated.

Zattoo employs an advertising based business model (through the use of banner ads, targeted text ads and inserted video clips) [158] as well as paid subscriptions [160].

# 5 Considerations

This P2P technology survey reveals that the field of P2P content delivery (typically in the non-commercial sector), is a very diverse one.

The general evolutive trend, initially, was towards growingly decentralized solutions. In time a return as occurred to more centrally coordinated modes of operation, (BitTorrent, FastTrack), in order to regain some operational efficiency in terms of content discovery and location and overall network connectivity.

In what concerns security, the majority of P2P solutions is very poor. Those including such provisions are not really robust. Provisions expressively designed and intended for the enforcement of content usage rights and the protection of copyright, are practically universally absent.

Commercial, P2P based, media delivering initiatives, generally employ either proprietary P2P solutions or pre-existing ones. The technical details of the earlier solutions are typically not disclosed. Still, from the few that have been disclosed, it may be concluded that such solutions employ some form of partially centralized operation.

The latter solutions are predominantly based on Gnutella. This is a purely distributed P2P protocol which is very robust, in terms of fault tolerance, but also considerably inefficient in terms of content discovery and distribution.

In the studied initiatives, the P2P delivery mode frequently performs a parallel, or even just, auxiliary role to client/server delivery mode.

These initiatives, have generally sought to maintain their business and legal (copyright related), operational legacy. This means that they have strived to maintain control over their content's usage throughout its lifecycle, that is, after it has been distributed. To attain such a control, (and given the lack of adequate security and copyright protection facilities, predominant in P2P content distribution platforms), they have typically teamed the P2P distribution systems with content protective DRM technologies.

Thus, most commercial P2P content delivery initiatives employ (or, at least, did so originally), classical DRM technology. Given the tasks it has had to perform, this technology focuses mostly, if not exclusively, in a restrictive governance of content usages.

The internal characteristics, operation and capabilities of the solutions specified by the initiatives presented in section 3.3.1 or implemented by the platforms described in section 3.3.2, may vary somewhat. However, overall, they are very similar in terms of their global operational logic and of their impact in content usage and usability, as their design was guided by the same requirement: content manipulation restriction throughout its lifecycle.

In P2P terminology, this DRM assistance, in security matters, (to the P2P delivery structure), may be described as a form of Trusted Third Party (TTP) solution. There is thus a visible decoupling of the content delivery structure from the security structure, where the latter performs the mentioned TTP role, and the potential synergies between the TTP structure and the P2P content delivery structure are left unexploited.

The typical resulting P2P+DRM structure may be depicted, albeit in a simplified manner, by Figure 22, (taking into consideration the notation defined in Table 1).

Its operation is as follows: At an initial moment (not displayed in the diagram), clients will authenticate with the TTP components agreeing on a common secret key ( $K_S^{session}$ ), which will be used for a client/TTP interaction session.

From that point on, all communication between a client and any TTP component will be encrypted with the agreed upon secret key (as represented by $\text{secmsg}_{K_S^{session}}(x)$). When the session terminates, the key is discarded.

Before a user (e.g. $u_A$) can be attended/hosted by a specific client/peer (e.g. $c_A$), he must first be authenticated. To do so, the user supplies his username and password, and $c_A$ packages it into $hsdAuthDat\,a_{u_A}$, by performing some hash function on that information (the case of employment of digest access authentication). The latter information package is sent (arrow 1) to the User Authentication Server (*UAuthS*). After validating the user's credentials, *UAuthS* sends $c_A$ (arrow 2), a user hosting certificate ( $S_{uhc}$ ), signed with *UAuthS* private key ( $K_{UAS}^{-1}$ ). $S_{uhc}$ proves that $c_A$ is hosting $u_A$. If a user (e.g. $u_A$) wishes to consume some media object (e.g. $o_A$), $c_A$ may retrieve it from the Content Server (client/server operation, arrow 3), or from another client/peer, such as $c_B$ (P2P operation, arrow 3a). The content is obtained in its protected form, ( $P_{o_A}$ ), which is encrypted with a secret symmetric key $K_S^{o_A}$ .

The client than contacts the License Server (*LS*), to inquire it about $u_a$'s usage rights over $o_A$, by sending it (arrow 5), $S_{uhc}$, (proving that $c_A$ is indeed hosting $u_A$), and the identification of the media object in question. The *LS* then responds (arrow 6), with the signed license ( $SL_{u_A}^{o_A}$ ) that specifies $u_A$'s rights over $o_A$ (assuming $u_A$ previously purchased such a license). $SL_{u_A}^{o_A}$ is signed by the *LS* so that its validity may be checked. $c_A$ then sends $SL_{u_A}^{o_A}$ (arrow 7) to the Key Server (*KS*). *KS* assesses the validity of $SL_{u_A}^{o_A}$ and the rights that it grants to $u_A$. If all is

ok, it returns (arrow 8), $o_A$'s decryption key ($K_S^{o_A}$). $c_A$ is then ready to access and render $o_A$ for $u_A$'s consumption.

Table 1 – Notation for Figure 22

| | |
|---|---|
| $u_i$ = a specific system user | $K_S^{session}$ = a comm session encryption key |
| $psswd_{u_i}$ = password of $u_i$ | $K_S^{o_A}$ = a $o_A$'s encryption/decryption key |
| $usname_{u_i}$ = the username of $u_i$ | $K_{LS}^{-1}$ = the private key of the License Server |
| $c_i$ = a specific system client | $K_{UAS}^{-1}$ = the private key of the User Authentication Server |
| $K_{c_i}$ = the public key of $c_i$ | $h(x)$ = cryptographic hash function applied to $x$ |
| $K_{c_i}^{-1}$ = the private key of $c_i$ | $enc_{K_{x_i}}(x)$ = the encryption of $x$ with $K_{x_i}$ |
| $o_i$ = a specific media object | $signed_K(x)$ = $enc_K(h(x))\|x$ = signed $x$ with key $K$ |
| $L_{u_i}^{o_i}$ = the license governing the use of $o_i$ by $u_i$ | $secmsg_K(x) = enc_K(x)$ |
| $K_S$ = a secret symmetric encryption key | |

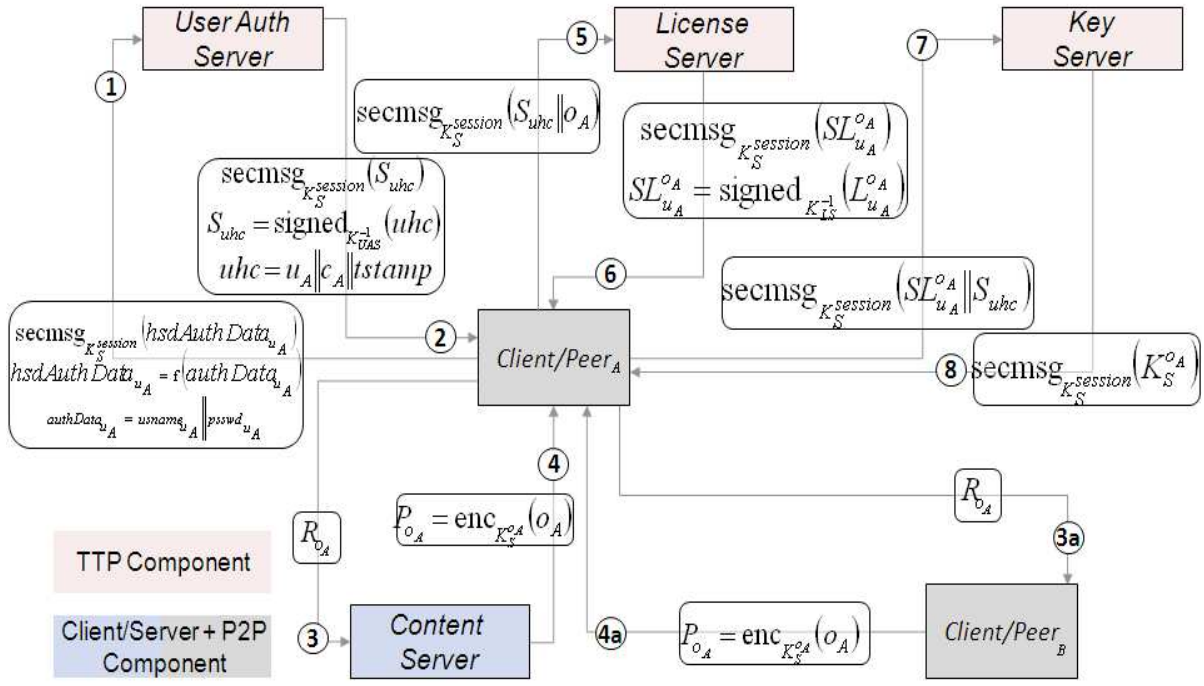


Figure 22 – Typical Commercial P2P Content Delivery System Architecture

In spite of their complexity and sophistication, the security services provided by the TTPs, have frequently been broken and circumvented by users [122], as this is incentivized by the uncomfortable content usage conditions that result from the operation of such security measures.

A currently observable trend, in the context of Commercial, P2P based, media delivering initiatives, is the abandonment of the DRM aspects of the platform, in favour of unprotected content. Also, the very P2P mode of distribution is also frequently being dropped in favour of a more traditional, (and less efficient), client/server mode.

In economic terms, the initiatives in scope have been trying a variety of BMs, such as subscription, purchase or advertisement based ones. They have frequently jumped between several BMs in their search for a suitable one and also frequently failed altogether. There is thus no globally predominant BM in this area and the attained economical results are also unremarkable.

For all of the above it is apparent that commercial employment of P2P content distribution, has been, and continues to be, considerably less successful that its non-commercial counterpart, at least in terms of its widespread employment.

The development of the earlier sector has been slow and hesitant. User captivation has been unremarkable, DRM provisions have frequently failed and P2P content delivery, in spite of its promises, has not yet been made to deliver on them.

# III Present Scenario Analysis

## 1 Introduction

In this chapter we examine the employment of the Internet for commercial content distribution, focusing on its present situation. That situation is examined so that its key problems and shortcomings are identified and a path to deal with such issues is delineated.
Section 2 thus carries a broad depiction of the present state of commercial on-line content delivery. Section 3 deepens the earlier description, analyses the described situation and identifies its subjacent causes. Section 4 further explores the causes of the problems currently facing commercial on-line content distribution, and defines the means by which they may be overcome.

## 2 Overview

The P2P content distribution technologies that have been developed have achieved significant success in the field of free content sharing. They include sufficient tools to enable a “good enough” content discovery and distribution capability. We describe them as “good enough” because it is satisfactory within a free content sharing context. However for commercial purposes, they can still benefit from performance improvements. This is so because, (as explained in section 2.4 of chapter II), the current typical P2P solutions are more or less distributed and typically have no provisions with anything close to a global systemic view. Lacking that oversight capability, the currently predominant P2P systems can provide for content discovery, locating and retrieval only through more or less distributed algorithms which are only of relative efficiency and consistency of results.

Furthermore, even if P2P content delivery systems/technologies include provisions to enable a minimally robust validation of content integrity (through checksum or signature calculation), they nonetheless present considerable shortcomings in the field of security and trust assurance, for even basic aspects, such as robust content integrity and authenticity assurance and data and communication confidentiality, integrity and non-repudiability.

On the other hand, the security-wise insufficiencies of the predominant P2P technologies can be effectively counterbalanced by the security-wise capabilities of existing DRM systems, if they are directed to that task.

The existing DRM technologies include robust tools to enable the, earlier mentioned, basic security measures. However, what has typically been demanded, from these systems, goes well beyond such requirements. They are asked to deliver, to content owners, almost complete control, over their content’s usage, throughout its lifecycle, regardless of said content’s location. Even so, DRM systems have managed to partially provide that, at the cost of imposing stifling content usage restrictions on consumers.

That provision was only partial, because, fuelled by consumer dissatisfaction, DRM schemes have been frequently broken or circumvented. After such events, they are then patched but only to be circumvented and re-patched at a later time, and so on [161], [162], [163].

Some shortcomings are thus visible, both in the current P2P content delivery and DRM technologies. Nonetheless, they are really not fundamentally hindering ones, especially considering the joint employment of the two technologies.

However, in spite of the proven results of the technologies in scope, and the success of non-commercial P2P content distribution platforms, the results of, commercial on-line content distribution initiatives (specifically P2P), have been dismal. This is demonstrated by the frequent cases of economical failure (e.g. Veoh or ReelTime), of change of activity area (e.g. Joost), and of the abandonment of P2P content exchange for a more traditional client/server one (e.g. BBC iPlayer).

# 3 Analysis

The current panorama in the field of P2P on-line content distribution from a business viewpoint (see section 5 of chapter II), is one of failure and of technological retreat (e.g. from P2P). However, this state of things does not fundamentally result from the technical insufficiencies of the involved technologies. It is the fruit of an historical process of technical evolution, which is changing the way we deal with information goods, as well as of the reaction to that evolution (on the part of the established CDists). In the next paragraphs we explain the mentioned process and associated reaction.

In the “Brick-and-Mortar” era, CDists controlled the content distribution structure (stores, CDs and DVDs production, etc). This structure was both socially useful and inescapable for consumer access to Information Goods (IGs). It was the scarcity and controllability, inherent to the materiality of the distribution structure that secured a useful and profitable role for CDists.

The Internet, given the automaticity and immateriality associated to it, is by far a more cost effective alternative, which eliminated the need for such a material structure. In this context, P2P content distribution technologies have played a crucial role, as they further reduce the costs of on-line content replication and distribution. Their employment and massive popularization, which started in the non-commercial sector, have thus experienced an explosive growth.

The main concern driving the development of (non-commercial) Internet-based content distribution initiatives was to take maximum advantage of the costs reduction possibilities offered by this medium. They thus focused on building distributed structures (typically P2P), capable of enabling a “good-enough” content discovery and an efficient content exchange between peripheral equipment. Guarantying content authenticity, copyright protection, or other security related aspects, was never the goal. These systems have thus often been connected to copyright infringing activities (copyright owners allege).

According to the Recording Industry Association of America, the music industry loses about $12.5 billion per year from all types of piracy activities [164]. The accuracy of these numbers is disputed [165], but it’s still plausible that considerable revenue losses have occurred as a result of the rapid increase in the free exchange of copyrighted content over the Internet.

Therefore, as the materiality of media content distribution is eroded, so is the control producers used to have over content use. All this is rendering the CDists’ traditional role, obsolete.

CDists have thus reacted by endorsing measures along two main courses of action. The first was legislative and punitive action against alleged copyright infringers. The second was the employment of technological measures, known as Digital Rights Management (DRM), to secure the control of the usage of their content throughout its lifecycle [166].
Simply put, the industry’s response has been to resist or attempt to reverse the on-going social and technological development with respect to the free flow of information.

In the legal sphere, CDists' efforts have led to the enactment of tighter intellectual property protection laws in some countries (for example, the Digital Millennium Copyright Act in the US). Furthermore, their frequent legal actions against alleged copyright infringers led to the closure of some content sharing initiatives (e.g. shutting down of P2P content sharing system Napster).

In the technical sphere, the content production and distribution industry has pushed hard for the adoption of DRM technologies, for the protection of all their content, regardless of the distribution medium. Furthermore CDists have also (allegedly), hired the services of some companies to mount poisoning attacks on P2P networks [125].

Nevertheless, CDists, even if reluctantly, have also begun to adopt on-line content distribution and to employ P2P technology. However, even though they are apparently moving in a progressive way, CDists are still trying to preserve their age-old reliable BMs. These initiatives have predominantly employed BMs which are based on (or close to) the direct sale of media goods (e.g. iTunes Store, 7digital, eMusic, Google Play, etc), and thus, depend on the restriction of access to such goods [167]. CDists are thus employing, apparently, technically progressive means but with regressive objectives.

The same applies to P2P commercial media initiatives that have started to appear, even though in an initial phase, alternative BMs were experimented with. Furthermore, given that P2P distribution was regarded as no more than an auxiliary aspect of their operation, these initiatives made no relevant efforts to enhance existing P2P structures. Already existing P2P structures (e.g. Gnutella), were, thus, simply used as they were. This typically meant that such structures were outside of CDists' control and presented unreliable performance.

In light of the above it can be concluded that CDists are holding on to BMs based on the exploitation of content scarcity in a medium, (the Internet), which is adverse to the preservation of such scarcity. To secure such BMs, DRM technologies are employed in an attempt to artificially maintain the necessary IGs scarcity.

The industry's overall conservative response has produced little or no results. In the legal field, the obtained results have failed to deter content sharing [5] [168], and have created considerable dissatisfaction from the consumer and small-creator/producer communities, causing costly damages to the industry's public perception [167].

In the technical field, end-user circumvention and rejection of DRM, has been a near constant, with digital piracy only increasing, despite the prevalence of new and increasingly elaborate DRM strategies [169].

The failure of DRM, to fulfil all that was demanded of it, is the result of the unrealistic level of that demand, and of an intrinsic vulnerability of DRM, which consists of the fact that users (the principal DRM violators), must inevitably be given access to the very content-encoding keys, whose secrecy is at the core of the operation of these systems.

As a result, all mainstream DRM systems (such as Apple's FairPlay, Microsoft's Windows Media DRM, and so forth) have been circumvented [161], [163]. In addition, DRM imposes usage restrictions that devalue the content and push consumers away. Finally, DRM isolates users because, in spite of much propagated interoperability, DRM systems typically lack interoperability [170], which means users are confined to the vendor's software and/or hardware.

DRM's failure means that the producers must still bear the fixed costs of information production, but their monetization schemes are losing efficiency. Their privileged foundation is thus, rapidly eroding and, the countermeasures employed by the producers have

backfired, earning them widespread animosity from the very people they are trying to win over as customers.

Also, these countermeasures have spurred the development of even more decentralized content-sharing systems (such as Gnutella), which are more resilient to legal and technical counteraction. In Michael Porter's terminology [171], these measures have merely built barriers to the entry of new legal competitors.

In this context, the overall results of on-line commercial content distribution (specifically P2P), have been unremarkable. In spite of the success of non-commercial P2P content distribution initiatives, the corresponding commercial counterparts have fared quite poorly, as shown by the frequent cases of economical failure (e.g. Veoh or ReelTime). Furthermore, the inevitable loss of control over content, that P2P content delivery implies, is in contradiction with the control levels required by the chosen BMs (by CDists). This is the reason why P2P content exchange has frequently been abandoned, in the initiatives exposed in section 4 of chapter II (e.g. Joost or Babelgum).

From a global perspective, commercial on-line content distribution (P2P or otherwise) is lagging far behind its once promised performance in terms of impact and adoption. Furthermore, under the pressure of the "free" content distribution, commercial distribution has frequently abandoned DRM altogether, in favour of unprotected content so as to offer a product which is not hampered by a lack of usability (which inevitably comes with DRM) in its competition with "pirate" or "free" content [172] [173].

The approach adopted by CDists has thus failed. Their control over information commodities continues to rapidly decrease and this is debasing the business methods that they still cling on to. In spite of all the potential of on-line content delivery (especially P2P based one), they have, thus far, failed to make it live up to its full potential, and the resulting panorama is, as described at the beginning of this section, one of failure and technological retreat.

Said failure fundamentally results, not from the above mentioned relative technical insufficiencies of P2P and DRM tools, but instead, as argued in [166], from the non-observation, on the part of CDists, that the Internet medium has come to radically alter the technical, economic and social premises underlining IGs production, distribution and consumption.

The low or negligible storage, reproduction and distribution costs that characterize this new medium mean that the occurrence of such activities, in mass, is inevitable. Content scarcity is, consequently, not a characteristic of the Internet. These factors and the failures of DRM show that such scarcity cannot be artificially imposed either. This medium's establishment, thus, represents the rise of a new technical and economical paradigm in the field of information goods' exchange and manipulation [166].

In such a paradigm information good scarcity is progressively annulled as they become pervasively accessible and manipulable. The access to and distribution of these goods is, thus, not something that may be restrained, and therefore, their monetization can no longer be approached in the same way of physically bound information commodities.

The immature state of development of commercial on-line content distribution, especially in the case of P2P employment, has thus resulted from CDists' anachronic attitude towards the on-going changes and towards the employment of this technology.

The reacquisition of a useful social role, by CDists and, consequently, the maturing of commercial on-line P2P content distribution, require that these entities accept and embrace

the changes brought by the Internet revolution. They must, thus, accept that, in the Internet medium, content scarcity is mostly a thing of the past.

# 4 Conclusion

Accepting the earlier conclusion, CDists must, therefore, replace the hitherto reliable value chain in which information commodities have been traded (on an access restrictive basis), with the employment of radical new BMs. These must not depend on content scarcity. They must enable CDists to relinquish having absolute control over their information goods and to make the most of the Internet's cost reduction capabilities.

For progress to occur, in the field of commercial P2P content delivery, it is thus, primarily, necessary that the mentioned new BMs are identified and validated. P2P content delivery technologies and associated DRM tools must then be made to support such BMs, by equipping them with any necessary capabilities.

The BMs, to be employed by commercial P2P content delivery initiatives, should typically be based on open content access. The extraction of gains should be performed in a lateral manner [166], which is more dependent on user voluntarism. This dependence means that the captivation and satisfaction of users will become even more important aspects then before.

To achieve such attractiveness and fidelization of users, the actual content exchange and consumption activities, in novel commercial P2P content delivery initiatives, should be inscribed in a virtual environment of socialization, which enables users to interact with each other and the media objects in secure, comfortable and reliable ways (i.e. social networks).

This demands that the platforms supporting these initiatives provide a content access service which is regular, consistent and reliable. They must thus assure content discovery, location and retrieval services, at a quality level equal or superior to that of centralized platforms (e.g. Google, in what regards content discovery and location). Furthermore, they must also assure peer and user identity, and data transaction security in order to guaranty communicational privacy and authenticity, and to assure monetary resource security. They must also assure media object integrity and authenticity.

As exposed throughout section 2 of chapter II (and summarized in section 2.4 of the same chapter), the predominant P2P networks and technologies present a number of insufficiencies, which meant that they are not yet at the operational reliability and security levels which are necessary to support the BMs in scope.

Current DRM technologies are powerful and may efficiently support a number of basic and sound security capabilities. However they are typically focussed on much more complex and encompassing objectives than the enforcement of the mentioned capabilities.

In light of the requirements that the platforms, supporting the initiatives in scope, must fulfil, it becomes evident that the state of the art in the involved technologies needs to be advanced. P2P content distribution systems/technologies must be equipped with the necessary tools for them to reach a higher level of overall operational constancy and reliability. These systems must also be given the capacity to assure peer and user identification, inter-peer communicational security, integrity and authenticity and to guaranty the integrity and authenticity of the distributed media objects.

The latter capabilities are within the responsibility scope of a DRM system. This means that such tools (DRM tools), should be added to P2P content distribution platforms. However,

DRM tools should have a different focus than their current “incarnation”. They should move from a situation where their key purpose is to enforce content access restriction, to a new one where rights management is interpreted in a more flexible way, and where their main goal is to ensure that security, reliability and trust pervade throughout the content exchange and consumption environment.

Summarizing: we may conclude that the specific problem that needs to be addressed to enable successful commercial, P2P based, content distribution, and hence, a safe distribution of rich media content, is to combine and appropriately adapt the different components that make up such systems, not just the technologies used.

The first component to be addressed are the BMs employed by commercial P2P content distributors, given the general inadequacy of the presently predominant ones. For that we must precisely assess the limitations of current BMs and identify the adequate BMs to overcome such limitations in a context of on-line content delivery. This is done in chapter IV.

Once this first component is solved, the second aspect that needs to be addressed consists of the operational and security shortcomings of present P2P technologies and of the current focus of DRM technologies. To address the earlier, a reliable and secure P2P infrastructure must be defined. The latter can be addressed through novel DRM/security tools that focus on providing support for security within the P2P infrastructure, whilst further assisting the previously identified business models, unlike the traditional tools, which focus on content access restrictions. This second aspect is addressed in chapter V.

Furthermore, the development of richer, semantically empowered and security supportive media object formats will also contribute to advancing the operational and security related capabilities of P2P technologies, at the same time providing additional value to the identified BMs. Addressing this aspect is, thus, also relevant.
Given the complementary nature of this later component, the manner in which it was addressed will not be exposed, in this dissertation, with the same level of detail as that of other components. The solutions devised in its scope will be laterally approached, as a part of the solutions devised to address the operational and security shortcomings of present P2P technologies, in chapter V. However, the contributions to the advancement of the state-of-the-art, which were achieved within this context, are presented in section 4 of chapter VI.

# IV Adequate BMs for the New Paradigm

## 1 Introduction

The Internet revolution and the exponential progress of the associated digital technologies, is causing a paradigm shift in the way we manipulate exchange and consume information/media content. These changes are now profound and deeply rooted in our Internet usage habits and expectations.

Reversing the occurred developments would require the enforcing of draconian legal and technical measures, which would, profoundly, impoverish the way we access and exchange information over the Internet. Furthermore, the development of social practices based on the free flow of information, that is currently happening, is a continuation of a longstanding trend that is shifting the socioeconomic support, for this field of human activity (intellectual goods production and exchange), from voluntary upstream support (patronage) to voluntary downstream support (user voluntary contribution).

To succeed, information goods producers and distributors should not fight these changes but instead embrace them by altering their operational paradigm. CDists need to realise that the controlling gatekeeper role that they occupied, for a very long time, was only a temporary one. It served its “purpose” of enabling the transitioning, of this field of activity, from the preindustrial patronage era to the information-industry era, and is now, considerably exhausted.

The role of the CDists, in the ensuing era, should be to maximize content distribution and access, to support captivating virtual social spaces, to promote artist and consumer bonding, and to foster voluntary consumer contribution to the sustainment of media content production.

In chapter III we exposed that the lack of adequacy of the BMs traditionally employed for on-line content distribution (P2P), stems precisely, from the fact that they generally go against the on-going changes. More specifically, traditional BMs depend on the maintenance of content scarcity in an environment (the Internet), which virtually eliminates such scarcity.

The new BMs, for on-line operation should not depend on the maintenance of content scarcity. As argued in [166], the scarce resource to be exploited, is no longer the information commodity but the user’s attention and fidelity. The earlier should be regarded as an investment that is made in order to obtain the latter. BMs should thus be based on open content access, enabling the free retrieval and consumption of information products by the user community, while exploiting the Internet’s capacity to eliminate costs in reproduction and distribution.

The actual extraction of gains should be performed in a lateral manner [166], predominantly, through the harnessing of user attention and good will. Accordingly, the BMs in scope, should seek to ensure that content manipulating activities take place, primarily in a CDist-managed virtual space of social interaction, where a culture of proximity and interdependency, (already on the rise), between consumers, artists and CDists can be fostered. This culture will then enable and sustain a collection of revenues based on voluntary funding and voluntary/accepted advertisement viewing by users.

BMs of this type, will, release CDists' technical infrastructures from the excess of content access control tasks that presently overburden them. These structures will thus be able to operate in much freer and more innovative ways, enabling commercial P2P content delivery initiatives to succeed and achieve the reliability that has, thus far, eluded them – content delivery reliability, content governing reliability and economical reliability.

# 2 Business Models

In light of what we have exposed previously, we have identified and defined the following BMs that are specifically suitable to support commercial, on-line. P2P based, content distribution, can be mainly identified as the following:

- user donation – in a donation-based business model, information/media items are voluntarily, and freely, delivered by artists, typically, expecting to be monetarily rewarded. Consumers will freely access such items and reward the creators they deem meritorious. In this context, CDists may obtain their revenue by taxing such donation transactions. Some real world examples where content production, and distribution, was/is supported by this BM (or a similar one), are, for instance, The Real News Network, Radiohead's and Nine Inch Nails' experiments with voluntary user donations or Wikipedia;

- ransom – in a ransom-based BM, the content producer (or the CDist on the producer's behalf), conditions the release of a specific content item to the prior reception of a specific amount of donations. Once that amount is reached, the content becomes available for all users, free of charge. In this context, again, CDists may obtain their revenue by taxing the donations;

- advertisement – in an advertisement-based business model, information/media items are voluntarily delivered by content producers. Users freely consume them, together with some advertisement messages. In this manner they supply their attention (advertisement viewing) in exchange for content access. That attention is sold to the advertisers for money, which is then employed to reward the content producers. In this context CDists typically obtain their revenue by retaining a portion of the amounts paid by the advertisers. This BM may take two different forms:
    - mandatory unrewarded advertisement viewing – in this mode the user is made to watch some commercial message before being given access to the desired content. The advertiser's payment is invisible to the user;
    - voluntary rewarded advertisement viewing – in this mode the user voluntarily proceeds to the consumption of advertisement messages. The value paid by the advertiser is shared between the CDist (which operates the user, advertiser and media interaction space), and the user.

These three BMs ultimately enable the free access to content and thus, do not depend on the existence or enforcement of content scarcity. This places fewer constraints on the management of content and enables a fuller exploitation of the Internet's cost reduction potentials. Those BMs are therefore appropriate for sustaining on-line content delivery initiatives.

These BMs do not represent a final and only choice for the Internet-era. They are the basic models. Others may, hypothetically, be developed, which are also inline with the guidelines of section 1 of the present chapter.

# 3 Validation

The validation of the proposed BMs is, first and foremost, a logical validation, based on the analysis of the technical characteristics of the Internet, (in the context of content distribution), and on the economic consequences of such characteristics. That logical backing was expressed in chapter III and approached again in section 1 of the present chapter.

To complement this validation an inquiry was performed to assess user receptivity to open media delivery and to the BMs described earlier. It was performed on a general population of users consisting of 88% content consumers and 12% content producers. The contents of the inquiry and the obtained results are presented in Annex C.

The inquiry was composed of four query groups. Such query groups had the following intents:

- Query group 1: evaluate user acceptance of the correctness of rewarding content producers and CDists; and assess user awareness of the fact that if such rewarding does not take place the content will cease to be produced;

- Query group 2: evaluate user receptivity to a total dematerialization of content distribution (i.e. distribution performed completely on-line);

- Query group 3: assess user general receptivity/preference relatively to content accessing modes supported though a lateral extraction of gains in opposition to those which imply a direct and mandatory payment for content access;

- Query group 4: evaluate user receptivity to three specific open content accessing modes based on lateral gain extraction. Such modes are: voluntary donation supported mode; advertisement viewing supported mode; and ransom supported mode.

From the obtained results the following facts, regarding user preferences and attitudes, may be concluded to be effective:

- the responses to query group 1 enable us to conclude that:
    - media users clearly acknowledge the justness of rewarding content producers and CDists;
    - media users recognize the need for such a reward, so that content continues to be produced, and acknowledge the unfairness of the lack of such a reward;

- the responses to query group 2 enable us to conclude that:
    - media users consider that on-line content delivery is more convenient, to them, than the traditional, physical distribution;
    - media users tend to accept that media goods be distributed, predominantly, on-line;

- the responses to query group 3 enable us to conclude that:
    - media users predominantly prefer models, of media content access on-line, which are based on indirect and/or voluntary payment over those which are based on direct payment for content access. This is so even if considering that they are to spend the same amount;

- the responses to query group 4 enable us to conclude that:

- media users are receptive to an open content access mode which is supported through a donation based BM, however most of them have not yet made any on-line donation;
- media users predominantly believe that a donation based BM, where they decide to whom donate, enables a fairer rewarding of content producers;
- media users are very receptive to an open content access mode supported through an advertisement based BM, especially so, in the modality where they are rewarded for consuming advertisement messages and may then employ such resources to reward content producers of their preference;
- users are also somewhat receptive to a ransom base BM, but to a smaller degree than previous models;

The verified facts regarding general user attitude and their receptivity to the content access modes, and associated BM, that we propose, mean that there is a clear conscience, on the part of users of the need and justness of rewarding content producers/distributors. Users reveal great receptivity for an open content access mode and show a clear willingness to cooperate in the economical supporting of content production/distribution by way of voluntary donations.

Media users are also very receptive to advertisement viewing and show some receptivity towards the payment of “ransoms” for content access.

These results give additional credibility to the BMs we have proposed.
Further backing of our views is provided by the acknowledgement of their value by scientific peers. This was attained by means of paper publications in scientific venues. This subject, however, is more extensively approached in sections 2 and 5 of chapter VI were we focus on exposing the effective novel contributions made by this PhD work.

# V A Reliable P2P Architecture for the New Paradigm

# 1 Introduction

In this chapter we describe a P2P content delivery architecture which is tailored to reliably support the BMs identified in chapter IV, which are fully inline with the emerging paradigm in media replication and distribution. To provide that support this architecture must address the requirements identified in section 2.1 of the present chapter.
The resulting architecture provides the BMs in scope with capabilities that the present state of the art in P2P technologies fails to do, in the fields of operational reliability and security assurance.

For ease of referencing, this architecture will be regarded as a system that shall be named P2PTube.

In the following sections, the precise requirements placed on P2PTube are identified, as well as the overall technical implications of supporting such requirements (section 2). Subsequently, the structure (section 3), data structure (section 4), and operation (section 5), of P2PTube are explained, as well as its inter-system cooperation capabilities (section 6). We then present the structure and content of the most relevant data objects, employed in P2PTube (section 7). Finally we explain how the system's base operational capabilities may be employed to support its commercial exploitation in accordance with the BMs identified in chapter IV (section 8).

# 2 Requirements and Implications

## 2.1 Requirements

### *2.1.1 Introduction*

The ultimate requirement placed on the P2PTube architecture is that it be able to provide the necessary tools to support the deployment of the BMs identified in chapter IV. It must therefore enable an efficient and open P2P content distribution and securely support a lateral extraction of gains.
The system's global operation should be integrated and seamless so that a fluid and rich inter-user and user-media interaction is possible, in order for a dynamic, robust and reliable interaction environment to be maintained, which facilitates social interaction and the offering of rewards.

These overall requirements, in their turn, imply a set of base technical requirements such as a safe and optimal operation. The next sections explore the requirements in scope, in greater detail.

### *2.1.2 Overall Requirements*

#### *2.1.2.1 Business Requirements*

The system operating (a CDist) entity and the content producers (if they so desire), should be rewarded for their work. The latter should be rewarded by means of user donations and

the earlier, by taxing such donations. In said donations, consumer users reward, producer users, with monetary amounts which the earlier injected into the system, or which they acquired by selling their attention (advertisement viewing).

The system must, thus, deliver the necessary provisions to enable the sale of consumer user attention and its purchase by advertiser users. It must also be the mediator of all rewarding transactions and tax such transactions.
Furthermore, the system must be capable of performing the monitoring of content usage activities, on the part of consumers, both on and off line, at the “client” side, so that value may be extracted from that information.

#### *2.1.2.2 Rights Protection Requirements*

The system should protect the authorship rights of content creators. It should assure the ineludible binding between each specific media item and the identity of its owner.

Any kind of rights usurpation or infraction should be impeded. No user may be able to claim ownership or collect rewards for an item which is not truly his. Abusive actions over contents, such as unauthorized changes or removals (by the original author), insertions or diffusions of corrupted versions or unauthorized copies (of items already in the system), should also be prevented or undone by the system.

#### *2.1.2.3 Usage Requirements*

##### *2.1.2.3.1 Usage Roles*

The system participating agents (users) should be able to access the system, in a secure and authenticated fashion, from any of the system’s peers, employing the same identity which must be globally recognized.

Said agents should have clear usage roles (or user roles), attributed to them so that their specific rights and duties and respected and upheld, in what regards interaction with both content and other users. Given that both professional and amateur content is desired, user roles should be accordingly defined.

Simple, or consumer, users are those who are mostly involved in the consumption of content and in the delivery of amateur information objects, for which they expect no reward.
Producing users, are those that supply the system with professional content for which they generally expect an economic reward.

One other type of actor in this system is the advertiser or sponsor. These actors insert into the system, information objects with advertisement contents. Advertisers pay the system’s operating entity for the exposure of users to their advertisements. It is up to the system to perform the exposition of individual users to advertisement messages.

##### *2.1.2.3.2 Content Usage*

Once adequately authenticated, system users should be allowed to search for, consume, insert, remove, comment or version information items in accordance with their specific access privileges (e.g. a specific user cannot remove an information item which is not his).

The searching of content should be simple, expeditious and actualized. The system should enable the user to perform searches over the totality of the system’s content.

#### *2.1.2.3.3 Monetary Resource Usage*

The system must provide the necessary technical and financial provisions to enable users to securely inject (into the system), donate and retrieve (from the system), monetary resources. It must thus securely maintain user accounts, where the users' monetary resources are stored. And enable users to seamlessly and securely access those resources.

### *2.1.3 Base Requirements*

#### *2.1.3.1 Operational Requirements*

The system should guaranty an efficient and smooth global operation in face of a transient, and sometimes sabotaging, peer population. It should assure such aspects as:

- overall scalability – the system should remain efficient and governable in the face of a growing amount of available content, of a growing number of participating users and of a growing number of linked peers;
- reliable content availability – every and any media item, stored in the system, should be, at all times, available for retrieval, by the peers;
- efficient and universal content discovery – content searches should be performed expeditiously and over the entire content repository;
- efficient content location – the placement of media items over the peer tissue should be robustly know or efficiently discovered whenever necessary;
- optimal content distribution – content should be diffused throughout the system's tissue in an optimal manner, minimizing the overall data transmission expenditure, distributing the workload homogeneously throughout the system and pre-emptively delivering content to areas where it is predicted to be desired;
- efficient content retrieval – content should be exchanged, between peers, in an efficient manner, so that its retrieval is expeditious and does not overburden any system peer;
- reliable content management – content insertion, removal, updating and versioning should be supported;
- credible enforcing of cooperative behaviour of peers – the system's peers should be compelled to maintain a cooperative and honest behaviour. Inadequate such behaviour should be detected and adequate measures taken to deal with it;

#### *2.1.3.2 Security Requirements*

##### *2.1.3.2.1 Secure Identification*

All system peers, users and media items should have a unique, global, verifiable and non-falsifiable identifier.

This identity should be employed:

- in the case of peers:
  - to identify the peer in every interaction it participates in, with other peers, so that peers are accountable for their actions, the permissions of each individual peer may be respected and so the fulfilment of their incumbents is verified;
- in the case of users:
  - to identify the user in every interaction it participates in, with the system, its content or other users, so that users are accountable for their actions, the rights of each individual user may be respected and so the fulfilment of their obligations verified;
- in the case of media items:

- to unequivocally identify the media item so that it may be universally referenced, and so that it is manipulated in accordance with its specific rights context;

The system is thus responsible for guarantying that the above mentioned identities are not misused, and for enabling their global accessibility and verifiability.

#### *2.1.3.2.2 Secure Communication*

The system must operate in manner which guarantees that all data exchanges between peers are confidential. It should preserve the integrity of said data, guaranty its undeniability and enable the validation of its authenticity.

#### *2.1.3.2.3 Secure Content*

The system must guaranty the integrity of all media items exchanged though its tissue. It must enable the validation of said items' authenticity and guarantee the undeniability of their authorship.

## 2.2 Architectural Implications

### *2.2.1 Introduction*

The satisfaction of the requirements, outlaid in section 2.1, has a number of specific, basic, technical implications on P2PTube's architecture, which are translatable into a set of architectural options. These are exposed in the next sub-sections.

### *2.2.2 Overall Implications*

The design of the overall architecture of P2PTube is primarily conditioned by the security requirements of the BMs to be supported, as the satisfaction of such needs is fundamental to guaranty the feasibility and sustainability of said BMs. The satisfaction of the operational requirements is conditioned to the satisfaction of the earlier.

For this reason at the base of the P2PTube architecture, providing the necessary security and trust, should be a PKI-like infrastructure, as these are robust means of guarantying security in distributed informatic systems.

Different types of PKIs (in a broad sense), exist. Such types and their most relevant characteristics are the following [58]:

- Centralized PKI – this is formed by a confederation of trusted entities, (called certification authorities, or CAs). These structures typically exist independently from other systems and merely provide security services to them. It is a tried and tested robust solution;

- Decentralized PKI – the PKI is maintained by the actual entities which need its services (the clients themselves). No central entities (CAs) are employed. This approach may be divided into the following three main sub types:

    - Web of trust based – in this scheme, every system node (peer) trusts (under the operating user's instructions), a specific set of other nodes, whose public keys it knows. Furthermore, every node relies on its trusted peers to certify the public key of yet other peers, and so on. Thus a web of trust is formed, where each peer may confidently discover the public key of another peer by finding a path, to it, in the peer acquaintance (trust) graph. This scheme

implicitly exploits the small world phenomenon of social acquaintance. Within that "small world" context it eliminates the need for central authorities. However, every trust chain is only as strong as its weakest link, and such links are typically subjective and transitive. Hence, this scheme, especially for large networks, is not without reliability issues;

- Statistical (quorum) based – in this scheme every peer's public key information is stored at multiple random other peers. This information is retrieved by obtaining a subset of these replicas from a subset of the peers. If a sufficient number of the peers behaves honestly it will be possible to securely obtain the public key. This scheme is thus dependent on the predominance of honest peers and implies an operational overhead in injecting, storing and retrieving the public key information;

- Hybrid – in a hybrid scheme, public key information replicas are stored and obtained from many peers. To validate the retrieved information, each replica is weighted against the trust the retrieving peer places on the replica supplying peer, and the global trust on the received information is calculated. This scheme is, potentially, more robust and efficient (in terms of information discovery), than the previous two alternatives, but is still short of reaching the robustness level of centralized PKI solutions.

In light of the above, and considering that, as argued in [48], some form of central certification authority is necessary to guaranty security in a distributed system, regardless of the honesty of the participating peer majority, the best security option, for P2P systems, is to employ some form of centralized PKI or similar security means.

As explained in section 2.4 of chapter II, P2P architectures have evolved to avoid having any coordinating central provision. This was done to avoid the costs of maintaining such an entity, but mostly, for legal reasons (avoid the legal targeting of that entity), and has resulted in operational shortcomings.

Many solutions and schemes have been devised, throughout time, to compensate for the lack of a coordinating entity, in P2P systems. However, these have never truly managed to provide the operational and security performances that central coordinating entities do.

The P2PTube architecture is specifically conceived to be employed by commercial and legally operating CDists, which derive the necessary revenue to sustain their activity. Such CDists can thus support the costs of maintaining said coordinating entities, and they are not at the risk of any legal attack.

There is thus a need for P2PTube architecture to have centralized coordination, in light of the security and operational demands that it must live up to. There is also a capability, of the P2PTube architecture to support such central provisions. For these reasons, the P2PTube architecture should therefore be equipped with central coordinating provisions which are to be responsible for the maintenance of trust and security, throughout its tissue, and for the coordination and optimization of that tissue's overall operation.

P2PTube's architecture should thus be a hybrid decentralized structure, which revisits and greatly enhances Napster's architecture, by fusing it with PKI-like capabilities. This architecture provides superior content manipulation and security capabilities because:

- its hybrid P2P structure – combines the security and coordinative capacities of a centralized system with the (reproduction and distribution), costs reduction properties of P2P content distribution;

- the integration of its robust security provisions with the hybrid content delivery structure – enables the exploitation of synergies between the two;

Furthermore, in order to empower the lateral extraction of gains, the system must also include provisions for monitoring user actions at the terminal side. These are also to be coordinated by the system's core.

The central part of the system, (the system core), should be the ultimate source of trust and security in the system. It thus is the ultimate source of identification information for all system entities, and of all other information, regarding such entities, as well. It is also to handle all tasks which require a global view of the system.

The peripheral part of the system (composed by the collective of user owned peers), is to employ a predominantly P2P mode of operation, under the coordination of the system centre.
It provides the users with an interface to the global system and its contents. Peers enable users to perform content searches, insertions, versionings and removals and a number of other user-system interactions.

### *2.2.3 Implications Upon the Core*

The satisfaction, of the previously specified requirements, has a number of technical and operational implications, specifically, upon the system core. Said core must thus:

- attribute, manage, deliver and guaranty the uniqueness and un-falsifiability, of the identities of the users, peers and media items involved in the system;
- collect, maintain and securely deliver public key information pertaining to all the entities in the system;
- collect, maintain and deliver information mapping peer identifiers to the corresponding IP address;
- function as the global registry, and ultimate validator, of media content and respective semantic and security related metadata. It must thus handle the original injection of media items, perform their backup storage and their original seeding. It must also prevent, or undo, the insertion of content whose authorship is incorrectly claimed;
- collect, maintain and deliver information regarding the location of media items over the peer tissue, and coordinate their retrieval from such locations, by content retrieving peers;
- enable and coordinate the updating, versioning and removal of media objects;
- support the execution of semantic content searches over all the content that the system makes available;
- maintain and manage user accounts and intermediate the transactions in which such accounts take part;
- attribute roles to system users and oversee their maintenance of such roles, and the actions they perform in the fulfilment of such roles;
- supply the necessary mechanisms to enable the exposing of users to advertisement content, the rewarding of users for the sale of their attention and the charging of the advertisers for that same fact;
- supply the necessary mechanisms to enable users to perform donations to producer users;
- coordinate the monitoring of user activities, at the terminal side, by the peripheral peers and collection of resulting information;
- handle the maintenance and coordination of "spaces" or means through which the interaction between users and content, takes place.

Given the operational and security demands placed upon the system's core, its constituting provisions should be maintained (economically), and managed by the, commercial, system operating entity. Only this way will it be possible for it to deliver the necessary trust, to the rest of the system, and maintain the necessary operational performance.

### *2.2.4 Implications Upon the Peripheral Peers*

The satisfaction, of the specified requirements, has several technical and operational implications, specifically, upon the system periphery, that is, upon the peripheral peers. A peripheral peer must thus:

- be able to securely, and in an authenticated manner, access the services of the system core and to answer its solicitations;
- be capable of securely, and in an authenticated manner, interacting with other peers by performing solicitations to them and by answering those peers' solicitations (e.g. participating in the inter-peer distribution of media items);
- follow the instructions of the system core regarding the servicing or shunning of other peers and their requests, so that a globally coherent strategy may be enforced for the curtailing of nefarious peer behaviour;
- comply to the monitoring requests received from the core;
- be able to validate media item integrity and authenticity, upon their retrieval, and to reliably store them;
- perform the interfacing between the users and the overall system. This means that the peer must possess the necessary provisions to:
    - enable the user to login to the system;
    - enable the user to perform global content searches, content insertions, removals, versionings and consumptions;
    - allow the user to enjoy the social interaction capabilities made available by the system;
    - enable users to manipulate the monetary resources stored in their accounts;

### *2.2.5 Summary*

Table 2 performs a summarizing mapping between the main identified requirements and their architectural implications.

Table 2 – Requirements to Implications Mapping

| Requirements | | Implications | | |
|---|---|---|---|---|
| | | Overall | On the Core<br>Core peer(s) | On the Periphery<br>Peripheral peer(s) |
| Business | Enable user donations | Hybrid decentralized structure | Manage user accounts<br><br>Intermediate all transactions | Enable users to securely interface the core services which enable them to inject money into their |

| | | | | |
|---|---|---|---|---|
| | Enable user attention sale | | Enable the secure insertion of money<br><br>Coordinate and intermediate advertisement exposure and the associated advertiser payment<br><br>Enable secure donations amongst users<br><br>Support secure extraction of money<br><br>Coordinate the monitoring of user activities, and collect the resulting information | accounts, donate, sell attention and extract money from their account<br><br>Perform the monitoring of user activity and the collection of related data at the request of the system core |
| | Tax user donations | | | |
| | Enable monetary withdrawals | | | |
| | Support user action monitoring | | | |
| Rights Protection | Prevent authorship theft | | Collect user reports on illegitimate content<br><br>Intermediate insertion of media objects preventing or undoing the insertion of content with illegitimate authorship declaration | Enable users to submit reports, to the system core, regarding suspected infringing content |
| **Usage** | Support different user roles | | Attribute different roles to users<br><br>Intermediate all relevant user requests managing them in accordance with the user's role | Enable users to access the services that the system provides for them, in accordance with their role |
| | Enable comprehensive search | | Maintain global registry of all media objects<br><br>Intermediate or coordinate searches so that they operate over the entire content set | Cooperate with other peripheral peers in their content searches in accordance to core instructions |
| Operational | Possess overall scalability | | Coordinate the retrieval of media objects<br><br>Coordinate the distribution of peripheral peer servicing responsibilities over the core tissue | Follow the core's instruction pertaining to the retrieval of media objects and to the solicitation of services from the core tissue |

| | | | | |
|---|---|---|---|---|
| | Maintain a reliable content availability | | Handle the original injection, backup storage and seeding of media objects<br><br>Maintain global registry of media objects, and their semantic metadata<br><br>Support semantic content searches over all the content<br><br>Manage and deliver media item location information<br><br>Coordinate and track media object distribution amongst peripheral peers<br><br>Coordinate media item retrieval<br><br>Intermediate and coordinate the updating, versioning and removal of media objects<br><br>Be the ultimate validator of media objects and their associated information | Adequately respond to service solicitations by other peers<br><br>Adequately store and redistribute the media objects that the core instructs it to redistribute<br><br>Assist other peripheral peers in semantic content searches<br><br>Assist other peripheral peers in the location of media items within the system's tissue<br><br>Be able to perform the insertion, search, location, retrieval, and removal solicitation of media objects, resorting to the assistance of the core or of other peripheral peers<br><br>Enable users to perform content searches, insertions, removals, versionings, retrievals and consumptions |
| | Enable an efficient and universal content discovery | | | |
| | Maintain an efficient content location | | | |
| | Enforce an optimal content distribution | | | |
| | Enable an efficient content retrieval | | | |
| | Enforce a reliable content management | | | |
| | Maintain a credible enforcing of cooperative behaviour of peers | | Coordinate, instruct and track peer cooperation<br><br>Collect peer reports on other peers' inadequate behaviour<br><br>Expel uncooperative or sabotaging peers | Report inadequate peer behaviour to the core<br><br>Follow the core's instructions pertaining to the shunning of peers and users |
| **Security** | Secure identification | | Maintain a global registry of all media object, user and peer identification or security information (e.g. public key information)<br><br>Attribute, manage and guaranty the validity of user, peer and media item identifiers and | Be able to securely authenticate and to establish secure communication sessions with the system core (login) and with other peers (inter-peer session)<br><br>Access the services of the system core and of other peers within the |

| | | | | |
|---|---|---|---|---|
| | Secure communication | | security credentials<br><br>Manage the mapping of peer IDs to their IP addresses<br><br>Enable the establishment of secure communication sessions for peers and users<br><br>Grant access to media object validation information | context of secure communication sessions<br><br>Enable users to establish secure communication sessions with the system (login)<br><br>Be able to validate media object integrity and authenticity (with the assistance of the system core and other peripheral peers), upon their retrieval, and to reliably store them<br><br>Submit reports, to the system core, if and when corrupted or invalid content is detected<br><br>Be able to adequately authenticate media objects at insertion time |
| | Secure content | | | |

# 3 Structure

## 3.1 Introduction

Section 3 presents the structure of the P2PTube system. Section 3.2 presents the system's overall structure. It begins by presenting that structure from a horizontal perspective, looking at the system as a whole, regardless of its vertical differentiation (layers), paying attention to the different types of peers that make up that system and, thus, constitute its horizontal differences. In the subsequent part of section 3.2 the structure in question is presented from a vertical perspective, therefore, focusing on its different layers and on the services that they provide each other. Section 3.3 presents the structure of individual peers and section 3.4 defines the structural roles that peers may play within the system.

## 3.2 Structure Overview

### *3.2.1 Horizontal (or Peer Based) Perspective*

The P2PTube's structure is composed by a collective of different types of peers organized in a hybrid decentralized architecture. That collective is composed of a single coordinating Central Core Peer (*CCP*), a group of coordination assisting Outer Core Peers (*OCPs*), and any number of, user hosting, Peripheral Peers (*pp*), as presented in Figure 23.

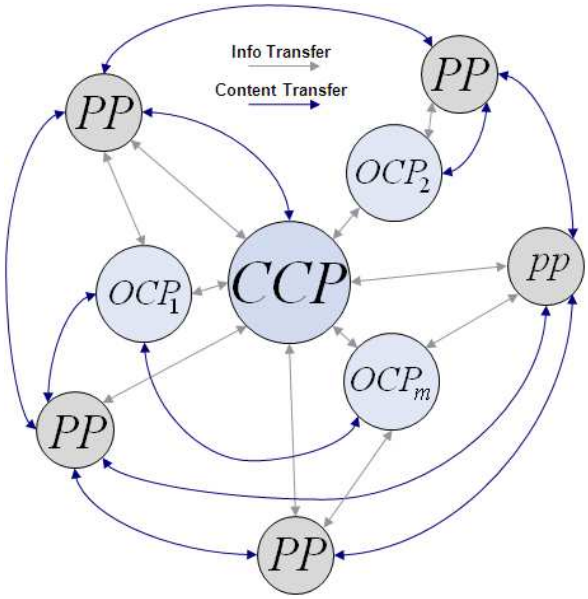


Figure 23 – P2PTube Structural Overview

The core section of the system's structure will be occupied by highly reliable and trustworthy peers, which are controlled by the system's operating entity and are always on-line. The system's peripheral section is more heterogeneous. It is composed by (regular user owned), peers with varying computational capabilities, communicational bandwidth, trustworthiness, reliability, and with an intermittent availability (as they connect and disconnect from the system). This section will thus present a very variable composition and topology.

### *3.2.2 Vertical (or Layer Based) Perspective*

The P2PTube's architecture is divided into horizontal layers. Each such layer provides services to the one above it and employs the services of the one bellow it.

More specifically, the P2PTube architecture is horizontally divided into the following layers:

- The Usage Environment Layer (UEL) – handles all aspects related to the interfacing of the user with the system and to the accommodation of the system core's user action monitoring requests;
- Peer Link Layer (PLL) – handles the resolution of peer contact endpoints and the securing of all inter-peer communication;
- Inter-Peer Communication Layer (IPCL) – handles the actual exchanging of communication messages between different peers.

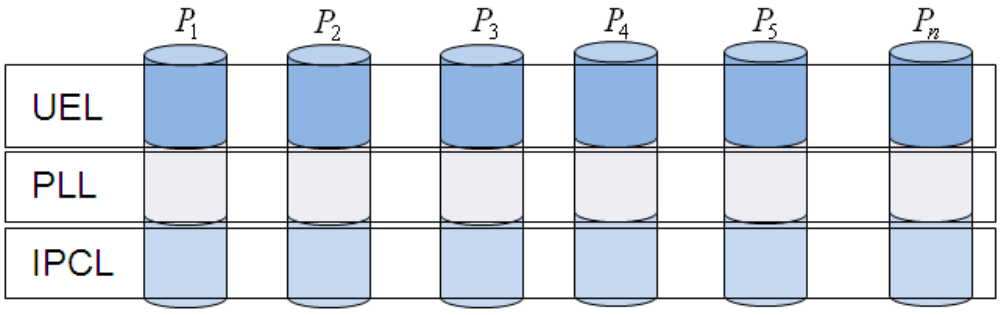


Figure 24 – P2PTube Horizontal Layers Traversing Peers

All three layers traverse all of the system's peers, that is, all peers are horizontally divided into their own particular instances of the UEL, PLL and IPCL. The P2PTube architecture may

thus be depicted, in what regards its horizontal division, in the manner presented in Figure 24.

## 3.3 Peer Structure

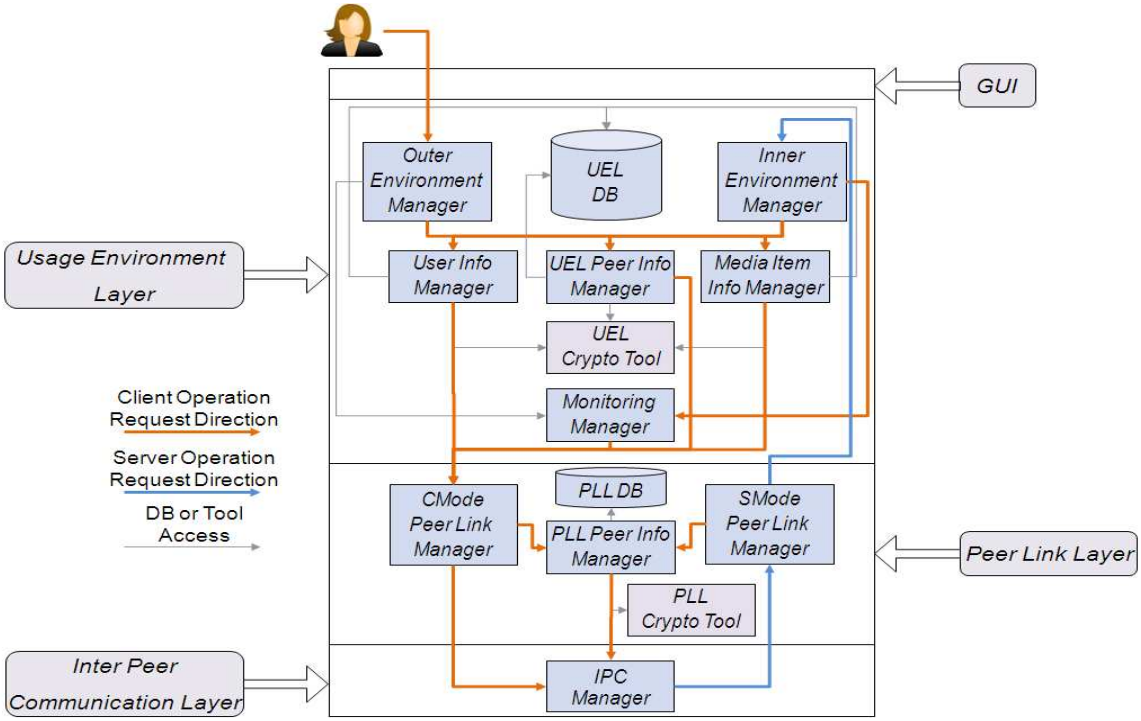


Figure 25 – Peer Architecture

All of the system's peers have the same internal structure (presented in Figure 25). As explained in the previous section, each peer's structure is divided into three layers:

- Inter-Peer Communication Layer – at each peer, this layer comprises the following component:
    - IPC Manager (IPCM) – receives internal, outbound, messages (from the top layers), and sends them to remote peers. Receives external, inbound, messages and sends them to the top layers;

- Peer Link Layer (PLL) – at each peer, this layer comprises the following components:
    - CMode PL Manager (CMPLM) – manages the client side of the PLL's operation. It receives outbound, request messages, from UEL, which it authenticates and encrypts. Obtains the IP address of the target remote peers and passes, that information, along with the processed message, to the IPCL. It also receives external, inbound, response messages. It deciphers and validates the authenticity of such messages and passes them to the top layers;
    - SMode PL Manager (SMPLM) – manages the server side of the PLL's operation. It receives inbound, request messages, (from remote peers), which it deciphers and validates. If the messages are destined to the UEL, it forwards them to the Inner Environment Manager. If not, the SMPLM processes the messages itself. It also authenticates and encrypts outbound response messages. At the *CCP* It handles the PLL monitoring and managing operations;
    - PLL Peer Info Manager (PLLPIM) – handles the manipulation of peer related information (e.g. peer identification, public key, IP), for PLL purposes;

    - PLL Crypto Tool (PLLCT) – handles all encryption, decryption and signature calculation activities at the PLL;
    - DB – the PLL Database;

- Usage Environment Layer (UEL) – at each peer, this layer comprises the following components:
    - Outer Environment Manager (OEM) – receives, and handles the servicing of, all the requests coming from the exterior of the local peer, that is, from the user. It does so resorting to the User Info Manager, the Media Item Info Manager, and the UEL Peer Info Manager;
    - Inner Environment Manager (IEM) – receives, and handles the servicing of, all the requests coming from the system's interior, that is, from other peers. It does so resorting to the User Info Manager, the Media Item Info Manager, the UEL Peer Info Manager and the Monitoring Manager. At the *CPP*, It also handles the UEL monitoring and managing operations;
    - User Info Manager (UIM) – handles all affairs related to the manipulation of user related information, such as user identification, user public key, user profile info, user media property info, user peer property info, user account and corresponding monetary resources info, etc. It also handles the authentication (signing) of data objects in behalf of users and validation of data objects origin authenticity (signature validation) in what regards to users;
    - Media Item Info Manager (MIIM) – handles all affairs related to the manipulation of media item related information, such as identification and authentication (signature) information, semantic characteristics information, location (on the system's tissue) information, ownership information, etc.;
    - UEL Peer Info Manager (UELPIM) – handles all affairs which involve the manipulation of peer related information (e.g. peer identification, peer ownership, etc.) for UEL purposes;
    - Monitoring Manager (MM) – handles all user action monitoring procedures. It registers event report requests, issued by the system core. The OEM informs the MM about every user action that takes place. The MM matches that information against the monitoring requests. If a match occurs a report is produced and sent to the system core;
    - UEL Crypto Tool (UELCT) – handles all encryption, decryption and signature calculation activities;
    - DB – is the UEL's database. It is employed by the UIM, the MIIM and the UELPIM. It stores information on users, peers and media items.

Given the P2P nature of the system, a peer may receive requests from the system's exterior (the user), or from its interior (other peers). The OEM thus handles exterior requests, which lead the peer to resort to the services of other peers, that is, to be a client of other peers; The IEM handles interior requests, which lead the peer to provide services to other peers, that is, to be a server to other peers;

## 3.4 Peer Structural Roles

### *3.4.1 Introduction*

The *CCP,* and all the *OCPs*, are to be owned and operated by the system's operating entity. All core peers run on high capacity hardware and have at their disposal large data transmission capacities. These peers are invariantly reliable and trustworthy.

The *PPs* are, typically, owned and operated by regular users, and, thus, commonly run on household PCs. These peers present varying degrees of reliability and trustworthiness.

The different types of peer's, with their different characteristics, present unequal levels of capacity, and trustworthiness. For this reason they perform roles with different levels of responsibility, authority and knowledge, at all of the system's layers.

### *3.4.2 Central Core Peer Role*

At all of the system's levels, the *CCP* is the ultimate authority in the system and should be fully trusted by all the peers. It is the base source of the system's integrated PKI-like capabilities. The *CCP* coordinates the system's entire operation and all the interactions that it comprises, for the servicing of the peripheral peer collective.

The *CCP* is the only peer in the system which contains the system's data structure in its entirety at all times. It is, thus, the ultimate source of all information, at all system levels, and the sole handler of all peripheral requests which imply some change or addition to the system's overall data structure.

As it is the root of all trust, in the system, the *CCP*, is, therefore, the ultimate assurer of the system's operational reliability, provider of the necessary functionalities for the maintenance of an environment of trust within the system, and deliverer of the necessary means to assure identity verifiability to all peers, users and media objects.

More specifically, the *CCP*'s responsibilities include:

- at the PLL:
    - storing the PLL part of the system's data structure (information on all peers), in its entirety;
    - monitoring the system's internal context, (by monitoring the requests received from the peripheral tissue and the information pertaining the interactions between the peripheral peers), in order to optimize the system's internal operation and the cooperation between its peers.
- at the UEL:
    - storing the UEL part of the system's data structure (information on all users and media items and the media items themselves), in its entirety;
- at both levels:
    - providing and/or coordinating (reading) access to the system's data structure, for *PPs* and *OCPs*;
    - providing the capability to, and controlling, the alteration the system's data structure;
    - guaranteeing the validity, coherence and correctness of the information contained in the systems data structure;

### *3.4.3 Outer Core Peer Role*

The *OCPs* assist the *CCP* in the servicing of the peripheral peer collective. Each *OCP* is a trustworthy alternative (to the *CCP*), for access to the system's data structure which they store, most of the time, in its entirety.

At the three system levels, the *OCPs* handle peripheral requests which imply only reading operations on the system's data structure.

Each *OCP* is, therefore, responsible for the delivery, of the information it stores, to the *PPs* or other *OCPs*. They are not, however, to service all and every *PP*. Whenever a *PP* logs into the system (comes on-line), the responsibility for handling its requests is attributed, by the

*CCP*, to one specific *OCP*. From that point on, the *PP*'s "reading" requests should be sent to the specified *OCP*.

There is thus a division of responsibilities, regarding the attending of the peripheral peer collective's requests, amongst the *OCPs*, which is coordinated by the *CCP*. Each *OCP* is responsible for the handling of the requests coming from a subsection of the set of all peripheral peers.

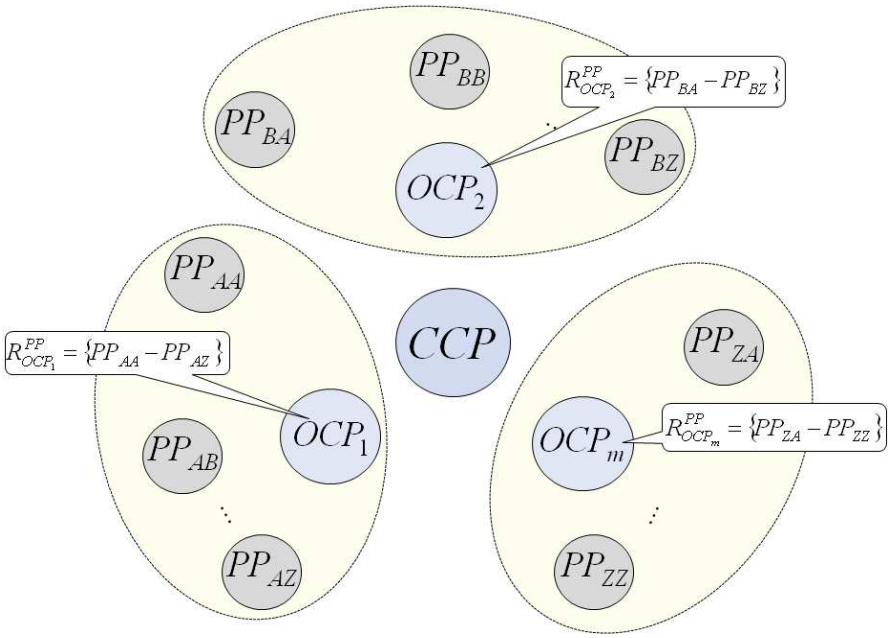


Figure 26 – Example Distribution of OCP Servicing Responsibilities over the PP Collective

Figure 26 presents a possible example of such a responsibility distribution. In it, the following assignments are in place:

- *OCP* 1 is responsible for servicing all "reading" requests from peripheral peers $PP_{AA}$ to $PP_{AZ}$;
- *OCP* 2 is responsible for servicing all "reading" requests from peripheral peers $PP_{BA}$ to $PP_{BZ}$;
- *OCP m* is responsible for servicing all "reading" requests from peripheral peers $PP_{ZA}$ to $PP_{ZZ}$;

For simplicity's sake the peer identifiers employed were only composed of two letters, thus enabling a total set of peripheral peers from $PP_{AA}$ to $PP_{ZZ}$. In reality this set will be much larger.

### *3.4.4 Peripheral Peer Role*

The *PPs* host the system users and service other peers in the system.

In specific terms the *PPs*:

- at the PLL:
  - attend to the requests coming from the UEL;
  - attend to requests, from other peers, for information on specific peers' information;
- at the UEL:

- perform the interfacing of users with the system by servicing their requests. This service typically implies the reading and/or righting of the system's data structure, which is achieved by resorting to the services of *CCP*, *OCPs* or of other peripheral peers;
- attend to monitoring requests from the system core;
- attend to requests, from other peers, for information on specific users or media items;

### *3.4.5 Summary*

Table 3 summarizes the main responsibilities of each type of peer.

Table 3 – Peer Responsibilities

| Responsibility Types | | Peer Type Responsibilities | | |
|---|---|---|---|---|
| | | Central Core Peer Resp. | Outer Core Peer Resp. | Peripheral Peer Resp. |
| ICPL Resp. | IPCL Manager Resp. | Receive incoming and send outgoing messages | | |
| PLL Resp. | PLL DB Resp. | Storing the entire PLL data structure at all times | Store all or part of the PLL data structure | Store core signed fragments of the PLL data structure |
| | PLL Peer Info Manager Resp. | Monitor PLL's overall operation<br><br>Provide and/or coordinate access to the PLL data structure, for PPs and OCPs<br><br>Provide the capability to, and control, the alteration of the PLL data structure<br><br>Guaranty the validity, coherence and correctness of the PLL data structure | Provide reading access to the PLL data structure to some specific subset of the PPs | Read and right to the PLL data structure (resorting to the CCP or OCPs), to perform its regular operation in service of the CMode or SMode PL Managers<br><br>Redistribute core signed fragments of the PLL data structure, to other, PPs, under CCP or OCP coordination |
| | CMode PL Manager Resp. | Handle outgoing requests (coming from the local UEL instance) and incoming request responses (coming from remote peers) | | |
| | SMode PL Manager Resp. | Handle incoming requests (coming from remote peers) and outgoing request responses | | |
| | PLL Crypto Tool Resp. | Perform encryption, decryption, signing, signature validation and hashing operations | | |
| UEL Resp. | UEL DB | Storing the entire UEL data structure at all times | Store all or part of the UEL data structure | Store core signed fragments of the UEL data structure |
| | Outer Environment Manager | Handle requests coming from the hosted users. Inform the Monitoring Manager about relevant user actions | | |
| | Inner Environment Manager | Handle requests coming from remote peers | | |

| | | | | |
|---|---|---|---|---|
| | User Info Manager | Monitor UEL's overall operation<br><br>Provide and/or coordinate access to the UEL data structure, for PPs and OCPs<br><br>Provide the capability to, and control, the alteration of the UEL data structure<br><br>Guaranty the validity, coherence and correctness of the UEL data | Provide reading access to the UEL data structure to some specific subset of the PPs | Read and right to the UEL data structure (resorting to the CCP or OCPs), to perform its regular operation in service the Environment Managers<br><br>Redistribute core signed fragments of the UEL data structure, to other, PPs, under CCP or OCP coordination |
| | UEL Peer Info Manager | | | |
| | Media Item Info Manager | | | |
| | UEL Crypto Tool | Perform encryption, decryption, signing, signature validation and hashing operations | | |
| | Monitoring Manager | Coordinate all monitoring procedures and registered the gathered information | Register event report requests, issued by the system core<br><br>Report to the core when relevant | Register event report requests, issued by the system core<br><br>Report to the core when relevant |

# 4 Data Structure and Model

## 4.1 Introduction

The set of all the information, regarding system participating entities (e.g. peers, information objects and users), system occurring events (e.g. media object insertions, consumption, inter user monetary donations, etc.), and system occurring relationships (the relationships between system entities and events, etc.), that the P2PTube architecture requires for its operation, collectively constitutes its data structure.

This structure is distributedly and redundantly stored throughout the system. It is redundantly held, in its entirety, by the *CCP* and the *OCPs*. The *PPs* store only fragments of it.

The data structure may logically be divided into UEL and PLL sections. However, operationally, it is one single coherent and interrelated structure.

The characteristics of data structure in scope, depend on the actual business model(s) supported and on the services to be delivered by this architecture. Said data structure must, thus, be able to support the description of realities of varying levels of complexity.

In the following sub-sections, we define the minimal necessary data model (to be embodied by the P2PTube's data structure), to support this architecture's operation and the business models outlined in chapter IV.

## 4.2 Rationale

Reality is made up of “things” which exist under involvement in relationships with other “things”. Each individual “thing”, in its turn, is also composed by interior “things”, which also exist under involvement in relationships with each other, and so forth, unlimitedly.

All of existence is transitory and is in permanent change. That way, with time, all things which had a beginning, as a relational aggregation of interior “things”, also have an end, that is, a disaggregation or alteration of the sub relationships and sub “things” that compose them. All things may thus be seen as events.

A car is an event composed by all of its parts and components under a specific set of interrelationships. If destroyed or disassembled, it ceases to be a car, that is, the car event is terminated. In the same manner, a storm is an event composed by air and water vapour masses maintaining a specific set of relationships amongst them and with the outer world. When those relationships change, or, for instance, the water vapour is removed from the storm (by way of the rain), the storm ceases to exist.

Relationships between “things” only exist when instantiated. That is, a relationship exists only when the two related “things” exist in a way so as to maintain it. This way, relationships are realized as occurrences, and are thus relational events. Relationships are therefore the most basic events. They are those which are closest to a mere logical existence.

Relational events consist of “predicates” and are thus connected to two other events of any type. One of these plays the role of the “subject” of the connection and the other one, the role of the “object” of the connection.

Oppositely to relational (or basic) events, the remaining events may be designated as complex events. These may be divided into two sub-types:

- Procedural events – these events consist of processes or actions (e.g. a flight, a collision). They are composed of only relational events and other procedural complex events. Ultimately, a procedural event composed of only one relational event is logically equivalent to it;
- Entitary events – these events consist of realities that may be seen as entities (e.g. a car, a person). They are composed by relational, procedural complex and entitary complex events.

The P2PTube system and its internal operation, as any other reality, may thus be described as a set of relational, procedural and entitary events.

For every reality a great number of entitary, procedural and relational events may be identified, at different levels of complexity. Depending on the context, not all of them, however, are of relevance to the representation or comprehension of the reality in scope. The aspects of reality that are to be effectively captured must be selected in accordance with the needs of every different situation.

In the specific context of P2PTube:

- the main complex events are:
    - entitary events: peers; users; and the information objects;
    - procedural events:
        - events pertaining to user interactions with the system, (e.g. user registration, insertion of media objects, donations);
        - events pertaining to internal, autonomously initiated, system activities;
- the main relational events are all those inter-webbing the tissue of complex events;

P2PTube's data model is therefore defined around these three basic types of phenomena and their most relevant instances. Given that procedural events represent complex occurrences in the system, they thus play the main structural role in the system's data model. The core of the model is therefore a continuous string of interrelated complex procedural events, which contain (or are somehow related to), a vast set of events of all types that describe the rest of the system's activity.

The richness and complexity of the data model, and its alignment with reality, makes it extremely versatile and, therefore, able to record any aspect of it, to any level of detail.

Traditional data models are conceived so as to perform a consistent registry of very specific aspects of certain domains. These models, are typically confined to registering a continuous succession of states in a given domain of reality, and, often, the registration of new states involves the loss of information about previous states. One might think of the information recorded in these structures, at any given moment, as a picture of the most relevant aspects of the state of reality, at that moment. At each relevant change, of the state of reality, so does the "photo" necessarily change, being that the changed aspect may, or may not, be recorded, in the same way that the context of alteration, in question, is generally not saved.

In an opposite manner, P2PTube's data model, does not record only the current state of reality. Instead, it registers the entire sequence of relevant procedural events, which occur, as well as their interrelationships, and thus, their context. It, therefore, captures a whole contextualized evolution of reality (and, thus, of its states) over time.
For example, in the case of a media item insertion, P2PTube's data model does not only register that said item is now available and some further details regarding it (e.g. its owner user). Instead, it registers the entire procedural event of the item's insertion, as well as its relationships with other relevant procedural and entitary events, such as the relationship with the procedural event consisting of the hosting of the owner user, at some peer, in the system and the events that consist of the hosting of the media item at various different peers, as a result of its insertion process. The information related to the availability of the media item or to the identity of its owner is present, all the same, but distributed across the recorded events and relationships. It comes, however, enriched by a whole, chronologically organized, context.

Given the way that P2PTube's data model is structured, it inevitably stores a more complex data structure. This means that retrieving information from it requires more complex operations. However, this is a trade-of, which enables, on the reverse side, an immense increase of the data model's versatility and of the wealth and contextualization of the recorded information. It also facilitates the continuous expansion of the model, so that the capture that it performs of reality, can evolve over time so as to meet ever-changing needs.

A more versatile, detailed and contextualized capture of reality, and with less information loss, means that more information, and more valuable information, is available to be accessed and exploited by both the system itself and by users. The system can thus, for example, correlate:

- a procedural event consisting of the insertion of a media item *A*;
- with a procedural event consisting of the hosting of its owner user at a given peer at given time interval;
- with a "*commenting*" relational event linking media item *A* with other media item *B*;
- with a procedural event corresponding to a consumption of a media item *C* in a given time interval;
- with a "*commenting*" relational event linking media item *C* and media item *B*;

and, therefore, determine that the referred user, inserted a comment to media item *B*, after having consumed a media Item *C* which also contained a comment to media item *B*, when he was hosted on a given peer at a given time interval.

Users, in turn, can make much more complex searches such as: what is the number of users that, after consuming a particular item *A*, consumed, immediately afterwards, a media item *B*.

## 4.3 Registered Information

The information registered by the system's provisions may be divided into two parts:

- The information that composes the global data model – this is the information which is permanently stored at the core data structure, and whose registry and knowledge is necessary for the system's global operations;
- The information that composes the individual peer data model – this is information which is stored at every peer, independently from each other, which translates the peer's individual experience and is necessary only for the proper operation of the actual peer.

For the sake of preventing the dissertation's main body of reaching an excessive size, the specification of the information which is registered in the system's data model is described in Annex D. In said annex the registered information is presented on an event-oriented basis that delivers a disassembled view of the overall resulting data structure.

The best way to understand the overall system's data structure, is to interpret it in an object oriented manner, as a complex mesh of interconnected events which is structured, and built around, a chain of procedural events in an hierarchical manner, starting from the core procedural events and then proceeding to all others.

Said core/structuring procedural events are:

- At the PLL – the *Peer Participation Session Events*;
- At the UEL – the *User Attending Session Events*;

From the above presented perspective the system's data structure consists of a series of events connected by relationships, where each event may "contain" (be bound to by *inclusion* relationships), relationships or further series of relationship connected events, and so forth.

Figure 27 presents a depiction of an exemplifying section of the data structure, from the above-mentioned perspective. In it, procedural and entitary complex events and entities are represented as green and purple boxes, interconnected by yellow relationship boxes.

From that image we may see that:

- the top level events are the *Peer Participation Session Events*. These can be seen as PLL events that consist of a peer being connected to the system for some time interval. *Peer Participation Session Events* are initiated, by *Peer Connection Events*, and terminated by *Peer Disconnection Events*, which in turn are triggered by *Peer-loginTo-System* and *Peer-loggoffFrom-System* relational events, respectively.

- *Peer Participation Session Events*, may include (be connected to, by *inclusionOf* relationships) other events, such as a *Peer Reporting Event*.

- *Peer Participation Session Events* are also bound, by *supportingOf* relationships to UEL levels events, such as the *User Attending Session Events*;

- *User Attending Session Events* consist of user participation sessions. They are initiated by *User Connection Events* and terminated by *User Disconnection Events*, which, in turn, are triggered by *User-loginTo-System* and *User-loggoffFrom-System* relational events, respectively;
- *User Attending Session Events* include all events pertaining to user interaction with the system. As such they may include events such as *MO Search Events*, *MO Responding Events*, *MO Commenting Events*, etc;

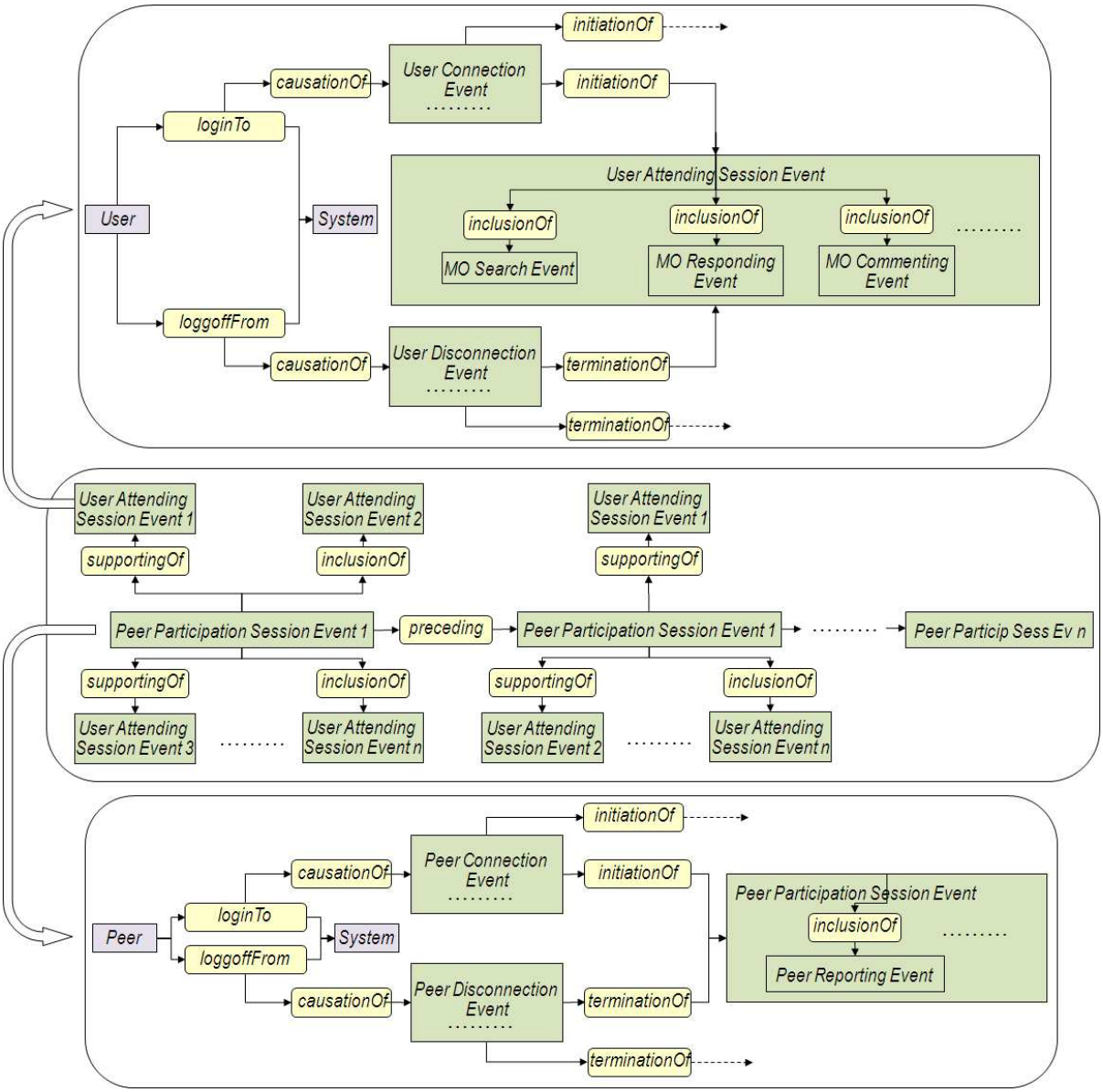


Figure 27 – Example of Data Structure Section

All mentioned procedural and entitary events are stored as entries in their corresponding tables, along with any other specific, relevant, information about them. They are webbed into the overall data tissue through their referral by the relational events that they are involved in.

All the explicitly mentioned relationships in the above text, as well as those implicitly defined, (when we say that some event includes some other event or relationship), are registered in the data structure as entries in the appropriate relational event table. In such entries there is

a field indicating the subject phenomenon, another referencing the object one and a third one indicating the "predicate" of the relationship, that is, the actual type of the relationship.

Furthermore, said entries, also carry an indication of the time instant at which the relationship "came into existence" and an indication of the time instant at which is became invalid.

As it can be concluded, the data structure will be an intricate mesh of information nodes (procedural and entitary events), interconnected by relational events, were no past information is ever lost. Procedural, entitary or relational events, which are no longer occurring or valid, have their end date introduced, and are marked as inactive. The present state of the system is described by the set of all the procedural, entitary and relational events that are still active. All other entries constitute the registry of the system's past.

## 4.4 Data Model

In accordance with the rationale presented in section 4.2 and the information set that the system's data structure needs to register (presented in section 4.3), three different types of aspects of reality (identified in section 4.2) are mapped into the system's data model:

- Entitary Complex Events – real or abstract "things" like peers, users and information objects;
- Procedural Complex Events – occurrences that take place within reality and are manifested through the initiation and/or termination of relationships;
- Relational Events – logical bonds between different segments of reality (e.g. between entities).

The data structure's model (presented in Figure 28), is defined in terms of those three aspects of reality.

All system events are registered at the *EVENT* table. This table carries the information fields, pertaining to events, which are common to all types of events. As such, each of its rows contain:

- EvID – the system wide unique identifier of its represented event;
- EvType – the specific type of the event (accepted values are RelEv, ProcCompEv and EntCompEv);
- BeggDate – the event's beginning time;
- EndDate – the event's end time (when available);
- IsActive – a boolean filed indicating if the event in scope is still effectively in occurrence and part of the system.

However, given that there are three main sub types of events, the latter table is "extended" by additional ones, in order to carry the information fields specific to each main type of event. This way:

- the *RELEV* table extends the *EVENT* table. It registers all Relational Events. Its entries carry the following fields:
    - RelEvID – the system wide unique identifier of the relational event (foreign key imported from the *EVENT* table);
    - RelType – the identification of the "predicate" of the relationship represented by the entry. It may take as value the name of any of the relational events defined in Annex D;
    - SubjectEvID – the identification of event which plays the role of "subject" in the relationship represented by the entry;
    - ObjectEvID – the identification of event which plays the role of "object" in the relationship represented by the entry;

- the *COMPEV* table extends the *EVENT* table. It registers all Complex Events. Its entries carry the following fields:
    - CompEvID – the system wide unique identifier of the complex event (foreign key imported from the *EVENT* table);
    - EvType – the identification of the type of complex event at stake. Said field may thus take as value either ProcCompEv or EntCompEv.

  This table is "extended" by the following ones:
    - *PROCCOMPEV* – this table registers all Procedural Complex Events. Its entries carry the following fields:
        - PocCompEvID – the system wide unique identifier of the procedural complex event (foreign key imported from the *COMPEV* table);
        - EvType – the identification of the type of procedural complex event at stake. It may take as value the name of any of the procedural complex events defined in Annex D;
    - *ENTCOMPEV* – this table registers all Entitary Complex Events. Its entries carry the following fields:
        - PocCompEvID – the system wide unique identifier of the entitary complex event (foreign key imported from the *COMPEV* table);
        - EvType – the identification of the type of entitary complex event at stake. It may take as value the name of any of the entitary complex events defined in Annex D.

All specific types of entitary, procedural and relational events whose registry requires the storage of some specific extra data field, besides the ones registered at their corresponding base tables (*ENCOMPEV*, *PROCCOMPEV* and *RELEV*), have their registry extended by specific tables, which store said extra information. Thus, if, for instance, the registry of entities of type "peer" needs to store some extra field, not available in the *ENTCOMPEV* table, then, for each registered peer, there will be an entry in the *ENTCOMPEV* table pertaining to said instance (peer). Furthermore, there will also be a *PEER* table (extension table), with a specific entry for the peer in question, which carries the additional, peer specific, information fields.

As it was implicitly shown above, in the extension tables, the primary key is a foreign key received from their corresponding base tables, which corresponds to the primary key of the latter tables. In the base tables, the field indicating the specific type of the event described in each entry, functions as a pointer to the specific extension table (if any), with further information about that event.

In more specific terms the basic data model contains the following extension tables:

- In the context of the registry of entitary complex events, the following tables extend the *ENTCOMPEV* table:
    - *PEER* – each table row contains peer-specific information pertaining to a specific peer;
    - *USER* – each table row contains user-specific information pertaining to a specific user;
    - *UACCOUNT* – each table row contains user-account-specific information pertaining to a specific user account;
    - *PEERIP* – each table row contains peer-IP-specific information pertaining to a specific peer IP address;
    - *IO* – each table row contains IO-specific information pertaining to a specific information object. This table is extended by the following ones, which pertain to different types of IOs:
        - *MO* – each table row contains MO-specific information pertaining to a specific media information object;

        - *MOFRAG* – each table row contains MO-fragment-specific information pertaining to a specific media object fragment;
        - *MORNSMANCMNT* – each table row contains MORansonAnnouncement-specific information pertaining to a specific MORansonAnnouncement;
        - *KEYPAIR* – each table row contains a specific private and public asymmetric key pair;
        - *OPERATIONALO* – each table row contains operational-IO-specific information pertaining to a specific operational information object. This table is extended by the following ones, which pertain to different types of operational IOs:
            - *SEARCHQIO* – each table row contains SearchQuery-IO-specific information pertaining to a specific SearchQIO object;
    - *SEMCONCEPT* – each table row contains semantic-concept-specific information pertaining to a specific semantic concept;
    - *SYSTEM* – each table row contains system-specific information pertaining to a specific system. In this particular case the table will carry one entry that pertains to the actual system that the model belongs to;
    - *TRUSTEDSYSTEM* – each table row contains system-specific information pertaining to a specific trusted outer system;

- In the context of the registry of procedural complex events, the following tables extend the *PROCCOMPEV* table:
    - *UDONATIONEV* – each table row contains user-donation-event-specific information pertaining to a specific user donation event;
    - *UATTSALEEV* – each table row contains user-attention-sale-event-specific information pertaining to a specific user attention sale event;
    - *UELOPEV* – each table row contains UEL-Operational-Event-specific information pertaining to a specific UEL Operational Event;
    - *UELCOOPEV* – each table row contains UEL-Coooperational-Event-specific information pertaining to a specific UEL Coooperational Event;
    - *UELMSGSENDEV* – each table row contains UEL-Message-Sending-Event-specific information pertaining to a specific UEL Message Sending Event;
    - *UELMSRECEPTEV* – each table row contains UEL-Message-Reception-Event-specific information pertaining to a specific UEL Message Reception Event;
    - *PLLCOOPEV* – each table row contains PLL-Coooperational-Event-specific information pertaining to a specific PLL Coooperational Event;
    - *PLLMSGSENDEV* – each table row contains PLL-Message-Sending-Event-specific information pertaining to a specific PLL Message Sending Event;
    - *PLLMSRECEPTEV* – each table row contains PLL-Message-Reception-Event-specific information pertaining to a specific PLL Message Reception Event;
    - *PCOMMSESSEV* – each table row contains peer-communication-session-event specific information pertaining to a specific peer communication session event;

- In the context of the registry of relational events, the following tables extend the *RELEV* table:
    - *UREWARDOFU_RELEV* – each table row contains *User-rewardingOf-User*-specific information pertaining to a specific *User-rewardingOf-User* relationship;
    - *UPAYOFU_RELEV* – each table row contains *User-paymentOf-User*-specific information pertaining to a specific *User-paymentOf-User* relationship;

- *UUNLOADOFUACCNTEV* – each table row contains *User-unloadingOf-User*-specific information pertaining to a specific *User-unloadingOf-User* relationship;
- *ULOADOFUACCNTEV* – each table row contains *User-loadingOf-User*-specific information pertaining to a specific *User-loadingOf-User* relationship;
- *STAXOF_UREWARDOFU_RELEV* – each table row contains *System-taxationOf-(User-rewardingOf-User)*-specific information pertaining to a specific *System-taxationOf-(User-rewardingOf-User)* relationship;
- *STAXOF_UPAYOFU_RELEV* – each table row contains *System-taxationOf-(User-paymentOf-User)*-specific information pertaining to a specific *System-taxationOf-(User-paymentOf-User)* relationship;

As it can be seen, the relational events tables play a crucial role in the interconnection of the data model. Each row/tuple in each such table represents one instance of a relationship. The information collectively carried by these tables describes relational events as *subject-predicate-object* constructs. It contains one foreign key for the subject event, one foreign key for the object event and whatever data fields are relevant to characterize the relationship.

This structural scheme of the data model grants it a great deal of versatility and makes it easily extensible, through the addition of extra event types (entitary, procedural and relational), and corresponding extension tables (if necessary), which build on the tables already present in the model.

## 4.5 Data Structure Distribution Over the System's Tissue

The system's data structure may be seen as having two scope-wise different sections. The main part is the global one (global scoped part of the data structure), which registers the events which describe the system's global operation. This knowledge is necessary to enable a proper global operation of the system, under the coordination of the core (*CCP* and *OCPs*).

At all system levels (PLL and UEL), the *CCP* and all the *OCPs* contain the core part of the data structure in its entirety. *PPs* contain only fragments of it.

However, given the system's distributed and P2P nature, this is not all the data in the system's collective data structure. In the course of their operation both the *OCPs* and the *PPs* will accumulate data that is not shared with the *CCP* (and does not need to be). This constitutes the locally scoped part of the data structure. For instance, they will register information on "*Peer Operation Events*", "*Peer Cooperation Events*" or "*Peer Communication Session Events*", pertaining to their regular operation and interactions with other peers, which is only relevant to them and does not need to be delivered to the *CCP*. This section of the data structure is thus particular to every peer and will not be stored at the *CCP*, thus being perishable.

The main access gateway to the global part of the data structure is the *CCP*. It intermediates all "writing" accesses to it, and takes care of timely updating the data structure copies located at the *OCPs*. The "reading" accesses to that section of the data structure are delegated by the *CCP* to the *OCPs*. These handle the distribution of the information and data objects contained in the system's data structure, throughout the *PPs*. As this distribution takes place, the *PPs* end up accumulating some knowledge of the information in scope (e.g. knowledge of Media Objects (MOs), answers to specific search queries, authentication information about specific peers or users, etc.), and the *OCPs* take note of this acquisition. In order to exploit that diffusion of knowledge, the *OCPs* distribute some of the *PP* servicing workload throughout the actual *PP* collective, by entrusting some *PPs* with the responsibility to service some specific requests of other *PPs* (see section 5.5.8.2.2).

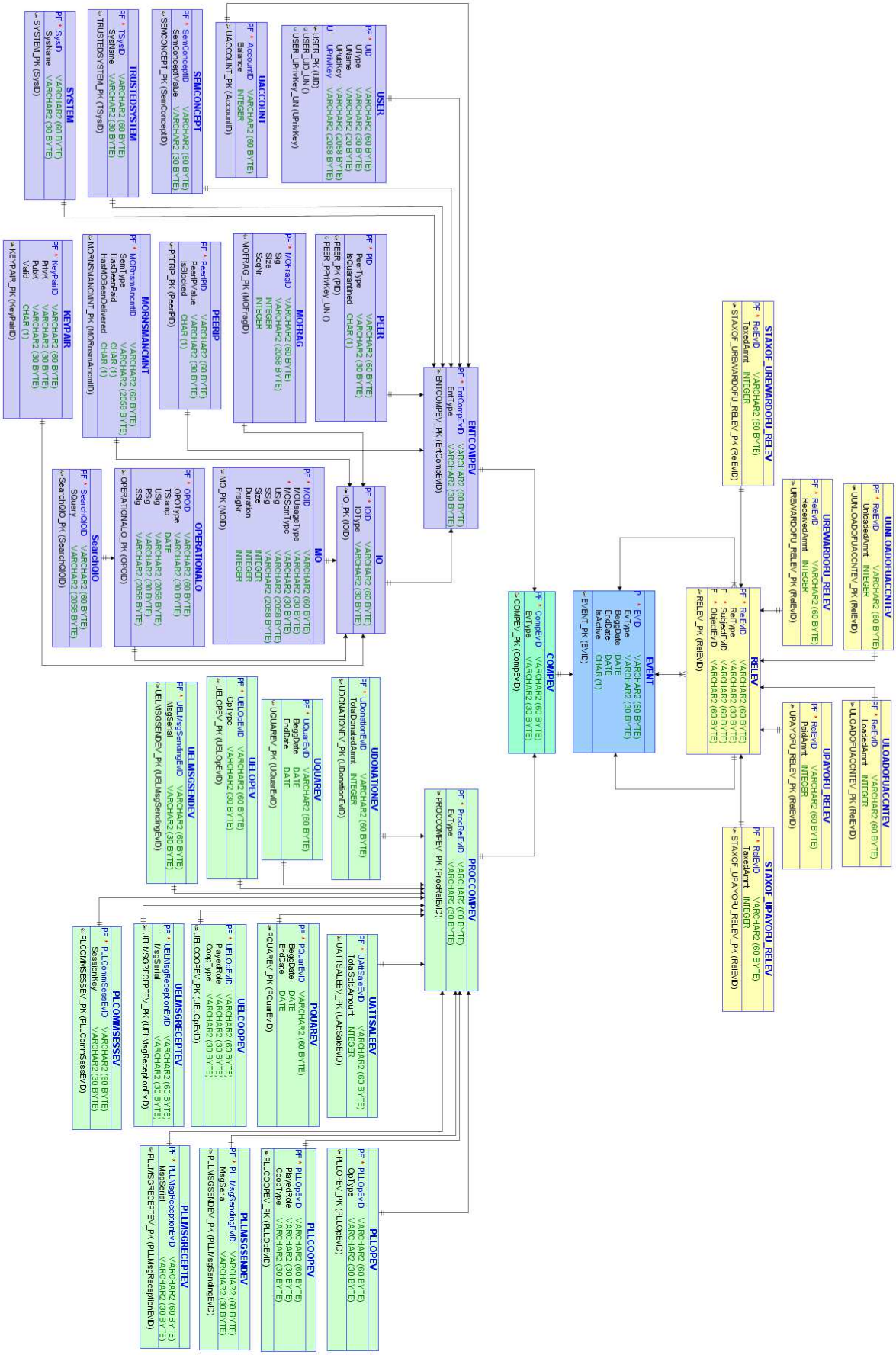

Figure 28 – Basic Data Model

This way, the system's data structure (or the relevant parts of it), is originally maintained and expanded by the *CCP*. It is then progressively diffused throughout the system's tissue, first to the outer core and then to the periphery. The responsibility of distributing that data (or parts of it), is also progressively entrusted to larger sections of the system, first to the outer core and then to the periphery.

# 5 Operation

## 5.1 Introduction

Section 5 explains the operation of the P2PTube architecture with a special focus on the security and reliability assuring aspects. In section 5.2 an overview is given of its main cross layer operational aspects. Section 5.3 presents the operation of the IPCL. Section 5.4 presents the operation of the PLL and section 5.5 presents the operation of the UEL.

In said sections the notation presented in Table 4 will be employed.

Table 4 – Notation Definition

| | |
|---|---|
| $U_i$ = a specific system user | $K_{P_i}$ = the public key of $P_i$ |
| $u_i$ = the id of $U_i$ | $k_{P_i}$ = the identifier of $K_{P_i}$ |
| $K_{u_i}$ = the public key of user $U_i$ | $K_{P_i}^{-1}$ = the private key of $P_i$ |
| $k_{u_i}$ = the id of $U_i$'s public key | $k_{P_i}^{-1}$ = the identifier of $K_{P_i}^{-1}$ |
| $K_{u_i}^{-1}$ = the private key of user $U_i$ | $MO_i^{ccp}$ = a specific media object |
| $k_{u_i}^{-1}$ = the id of $U_i$'s private key | $mo_i^{CCP}$ = the id of $MO_i^{ccp}$ |
| $P_i$ = a specific peer | $L_{u_i}^{MO_i^{CCP}}$ = the license governing the use of $MO_i^{CCP}$ by $U_i$ |
| $p_i$ = the id of $P_i$ | $K_{s_g}^{P_i-CCP}$ = a secret symmetric key for the encryption of inter-peer communication during the $g^{th}$ cooperation session between $P_i$ and $CCP$, $(s_g^{P_i-CCP})$ |
| $PP_i$ = a specific peripheral peer | $\mathrm{h}(x)$ = cryptographic hash function applied to $x$ |
| $pp_i$ = the id of $PP_i$ | $\mathrm{enc}_K(x)$ = the encryption of $x$ with $K$ |
| $OCP_i$ = a specific outer core peer | $\mathrm{signature}_K(x) = \mathrm{enc}_K(h(x))$ = signature of $x$ with key $K$ |
| $ocp_i$ = the ID of $OCP_i$ | $\mathrm{signed}_K(x) = \mathrm{signature}_K(x) \| x$ = signed $x$ with key $K$ |
| $CCP$ = the central core peer | $\mathrm{secmsg}_K(x) = \mathrm{enc}_K(x)$ |
| $ccp$ = the id of $CCP$ | |

## 5.2 Cross Layer Aspects

### *5.2.1 Base Operation Types*

From a vertical perspective, the system's overall operation is composed by the overall operation of each of its layers. The overall operation of each layer is composed by a myriad of different individual operations that concurrently take place within the layer's tissue. Each

such individual operation is composed of multiple interactions/cooperations between homologous layer instances of different peers.

From a horizontal perspective, the system's overall operation is composed by the overall operation of each of its peer. The overall operation of each peer is composed by a myriad of different individual operations that concurrently take place within the peer. As a peer is divided into different layers, so are the operations that occur within it. Thus, an individual peer operation typically has a UEL a PLL and an IPCL component.

If the two perspectives are integrated an individual system operation may be seen as a vertically integrated construct with the typical following structure:

- UEL level operation – composed of spanning tree of inter-peer interactions at the UEL level. At the root of that tree, is an inter-UEL interaction which has as its client the UEL instance at the peer which is triggering the entire operation. All other inter-UEL interactions occur in service to that originating one;
- PLL level operation – composed of multiple spanning trees of inter-peer interactions at the PLL level. At the root of each such tree, is an inter-PLL interaction which exists to service a superior inter-UEL interaction. All other inter-PLL interactions, in the same tree, occur in service to that originating one;
- IPCL level operation – composed of multiple inter-peer interactions at the IPCL level. Each such inter-IPCL interaction exists to service a superior inter-PLL interaction;

Individual system operations may be triggered by the attending of user requests or by the issuing of requests or instructions by the *CCP*.

An operation, originated by a user request, will cause a system-wide intra-UEL operation (as his request is picked up by the UEL instance at his hosting peer, which will typically resort to the UEL instances in other peers). This UEL level operation (set of inter-UEL interactions) will cause a PLL level operation (set of sets of inter-PLL interactions), which, in its turn will cause an IPCL operation (set of sets of inter-IPCL interactions). Therefore, a user-originated operation will be composed of sub operations occurring at each of the system's layers.

The *CCP* may trigger operations:

- at the UEL:
  - to request, from an *OCP* the updating of its copy of the system's data structure, pertaining to the UEL;
  - to request from a *PP* the monitoring of some user actions;
  - etc;
- at the PLL:
  - to request, from an *OCP* the updating of its copy of the system's data structure, pertaining to the PLL;
  - etc;

Operations triggered, by the *CCP*, at the UEL or PLL will also imply the occurrence of system-wide operations at the inferior layers.

The possible types of inter-peer interactions at every layer, composing the above-described operations, are the following:

- *PP*-to-*PP* – in this type of interaction a peripheral peer resorts to the services of another *PP*. Such an interaction may typically be considered a P2P interaction as it occurs between hierarchically equal peers and is bidirectional (what $PP_A$ may request from $PP_B$ may also be requested by the latter from the earlier);

- *PP*-to-*OCP* – in this type of interaction a peripheral peer resorts to the services of an *OCP*. Such an interaction may typically be considered a client/server interaction as it occurs between hierarchically different peers and is not bidirectional (what $PP_A$ may request from $OCP_B$, typically cannot be requested by the latter from the earlier);

- *PP*-to-*CCP* – in this type of interaction a peripheral peer resorts to the services of the *CCP*. Such an interaction may typically be considered a client/server interaction, given the hierarchical difference between the peers and the fact that the information that a $PP_A$ may require from $CCP$ cannot also be delivered from the earlier to the latter;

- *OCP*-to-*OCP* – in this type of interaction an outer core peer resorts to the services of another *OCP*. Such an interaction may typically be considered a P2P interaction, given the hierarchical equality between the participating peers and the bi-directionality of the involved types of requests;

- *OCP*-to-*CCP* – in this type of interaction an outer core peer resorts to the services of the *CCP*. Such an interaction may typically be considered a client/server interaction, given the broad hierarchical inequality between the participating peers and the lack of bidirectionality of the involved types of requests;

- *CCP*-to-*OCP* – in this type of interaction the *CCP* requests something from an *OCP* (e.g. the updating of the information that the OCP is storing). Such an interaction may be considered a client/server interaction, given the broad hierarchical inequality between the participating peers and the fact that the latter cannot request the same from the earlier;

- *CCP*-to-*PP* – in this type of interaction the *CCP* requests something from a *PP* (e.g. the monitoring and reporting of some user activity). Such an interaction may be considered a client/server interaction, given the broad hierarchical inequality between the participating peers and the fact that the latter cannot request the same from the earlier;

Depending on the specific interactions that compose an operation, it may be predominately client/server or P2P.

In terms of the involved peer types, the main types of inter-peer interaction sets, (operations) that will take place in the system, regardless of the level, are presented in Figure 29. Such sets are the following:

- User triggered interaction sets:

  - Type 1 interaction sets – a user submits a request to its hosting *PP*. If the user's request implies only the reading of (some part of) the system's overall data structure, the handling *PP* directs the adequate request to the appropriate *OCP*. For instance, if $R_{OCP_2}^{PP} \subset PP_1$ (as is exemplified in Figure 26), and $PP_1$ needs some information, it will direct a request, (for the needed information), to $OCP_2$. The contacted *OCP* will respond with the requested information;

- Type 2 interaction sets – this interaction set is very similar to the type 1 interaction set. The difference is that the contacted *OCP*,($OCP_3$), is for some reason, overburdened and thus redirects the $PP$ to some other $OCP$;

- Type 3 interaction sets – a user submits a request to its hosting *PP*. If the user's request implies only the reading of (a part of) the system's overall data structure, but the request (or some of the sub-procedures it implies) needs to be registered, the inter-peer interaction sequence will be similar to a type 1 interaction, except for the fact, that the contacted *OCP* will notify the *CCP* of the request received from the *PP*;

- Type 4 interaction sets – a user submits a request to its hosting *PP*. If the user's request implies some change or addition to the system's global data structure, the *PP* will have to direct the request to the *CCP*, as the latter is the overall maintainer of the system's global data structure and the gatekeeper of that structure in what regards changes made to it. In this situation (exemplified in boxed area 4 of Figure 29), the *PP* will then send the request to *CCP*. The latter evaluates its validity and if everything is ok, it performs the necessary changes to the data structure and sends a confirmation to the calling *PP*. In a parallel manner, given the (redundant), distribution of the data structure over the *OCPs*, the *CCP* will need to update the affected information in all the *OCPs*. It will thus issue update requests to all of them, with the new information, which these will store (exemplified by steps 2' and 3 in boxed area 4 of Figure 29);

- Type 5 interaction sets – this interaction set is very similar to the type 1 interaction set. The difference (as exemplified in boxed area 5 of Figure 29), is that the contacted *OCP*, $OCP_2$, hybridizes the operation by redirecting the original requesting *PP* ($PP_1$) to another PP ($PP_2$);

- *CCP* triggered interaction sets:

  - Type 6 interaction sets – due to the occurrence of some change or addition to the system's data structure, the *CCP* instructs the *OCPs* to update the data that they are storing (as exemplified in boxed area 6 of Figure 29, for a single *OCP*);

  - Type 7 interaction sets – the *CCP* needs some monitoring information to be obtained by some *PP*. In that regard (as exemplified in boxed area 7 of Figure 29), it sends a monitoring request to the target *PP* (step 1). The latter sends an acknowledgement and proceeds to perform the capture of the desired information. Whenever relevant, the *PP* sends back to the *CCP*, event reports containing its monitoring activity results.

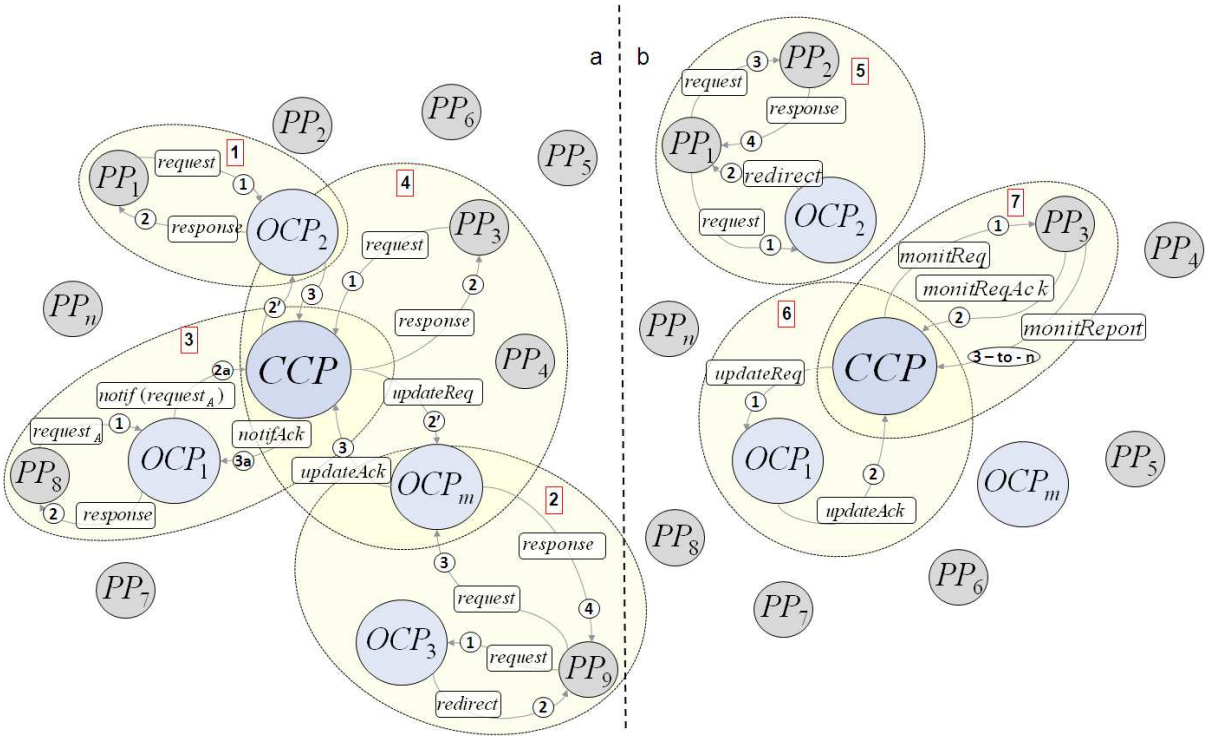


Figure 29 – Basic Inter-Peer Interaction Sequences

## *5.2.2 Operation Identification*

In the course of the system's operation, each of its peers will need to process numerous simultaneous operations, handling both user requests and the requests from other peers. This results in an overall scenario characterized by an intricate and concurrent set of global operations, where each such operation "traverses" multiple peers on multiple layers of the system, as it is composed of numerous inter-peer cooperations at multiple levels.
As all cooperations are performed through the exchanging of messages between the cooperating peers, the above-mentioned scenario translates into a complex and concurrent mesh of inter-peer message exchanges.

For the system to operate coherently it is necessary that no inadequate cross-communication occurs, that is, that the procedures and the exchanged messages of different operations and cooperations are not confused or mixed up. Operations must thus be clearly distinguishable from each other, and so must their internal composing inter-peer interactions, and, hence, their supporting messages.

In light of the above, operations must be attributed a system-wide unique identifier at the time of their initiation. To do so, whenever a peer (be it a *PP*, an *OCP* or the *CCP*), initiates an operation it builds a unique identifier for it, composed of the following parts:

- the operation initiating peer's identifier (which is *per se* system-wide unique);
- the operation's initiating time;
- the operation's sequence number, within the context of the peer and of the time of its initiation. For instance, an operation which is the 12$^{th}$ operation to be initiated by the peer (at some specific level, e.g. UEL), since the beginning of the last millisecond, will have "12" as its serial identifier;

For instance, if a *PP* is identified as "*p2ptube:ppeer:xpto001"*, if the current time in milliseconds is "*1341227480331"*, and if the operation is the "*12$^{th}$"* to be initiated by the peer in the current millisecond, than the operation's id is "*p2ptube:ppeer:xpto001:1341227480331:12".*

In order to correctly perform the operations that it starts, a peer will typically have to resort to the services of other peers, i.e., establish cooperations with them. These must also be uniquely identified. To do so, whenever a peer initiates an inter-peer cooperation, in the service of a specific operation, it will produce a unique identifier for it, composed of the following parts:

- the identifier of the encompassing operation;
- the identifier of the target cooperating peer;
- the cooperation's sequence number, within the context of its encompassing operation and of the target cooperating peer. For instance, a cooperation which is the 2nd cooperation to be initiated within the context (at the service) of the same operation, with the same target cooperating peer, will have "2" as its serial identifier;

For instance, if an operation is identified as "*p2ptube:ppeer:xpto001:1341227480331:12*", if the target cooperating peer is identified as "*p2ptube:ppeer:abc1"*, and if the cooperation is the "*2nd"* to be initiated in the context of the above mentioned operation with the above mentioned peer, than the cooperation's id is
"*p2ptube:ppeer:xpto001:1341227480331:12:p2ptube:ppeer:abc1:2".*

Peers may also become involved in operations that they did not start by servicing cooperation requests coming from other peers. In order to handle such cooperations, said peers may need to initiate further cooperations with other peers, and will, thus, need to adequately identify such new cooperations. To do so, whenever a peer initiates an inter-peer cooperation, in the service of a specific another cooperation (where it plays the server role), it will produce a unique identifier for the new cooperation, composed of the following parts:

- the identifier of the cooperation request that the new cooperation is servicing, and which is, thus, its originating cooperation;
- the identifier of the target cooperating peer;
- the cooperation's sequence number, within the context of its originating cooperation and of the target cooperating peer. For instance, a cooperation which is the 3rd cooperation to be initiated within the context (at the service) of the same originating cooperation, with the same target cooperating peer, will have "3" as its serial identifier;

For instance, if an originating cooperation is identified as
"*p2ptube:ppeer:xpto001:1341227480331:12:p2ptube:ppeer:abc1:2*", if the target cooperating peer is identified as "*p2ptube:ppeer:abc2"*, and if the cooperation is the "3rd" to be initiated in the context of the above mentioned cooperation with the above mentioned peer, than the cooperation's id is:
"*p2ptube:ppeer:xpto001:1341227480331:12:p2ptube:ppeer:abc1:2:p2ptube:ppeer:abc2:3".*

As stated earlier, all cooperations are performed through the exchanging of messages between the cooperating peers. These messages must be properly exchanged and managed to ensure the correct execution of operations and cooperations. Thus, the operation and cooperation identification scheme, presented before, is to be employed in the identification of said messages.

Every cooperation comprises the bidirectional exchange of one or more messages between two peers (the client peer and the server peer). The identifiers used in those messages are composed by the following parts:

- cooperation ID – the identifier of the comprising cooperation;
- cooperation side – a string which indicates if the message is coming from the client or server side of the interaction. In the earlier case its value is "*req*". In the latter case said value is "*resp*";

- counter – a numeric value indicating the cumulative number of messages sent from the message's sender peer to its receiver peer, in the context of the cooperation in scope. It is thus a counter of such messages.

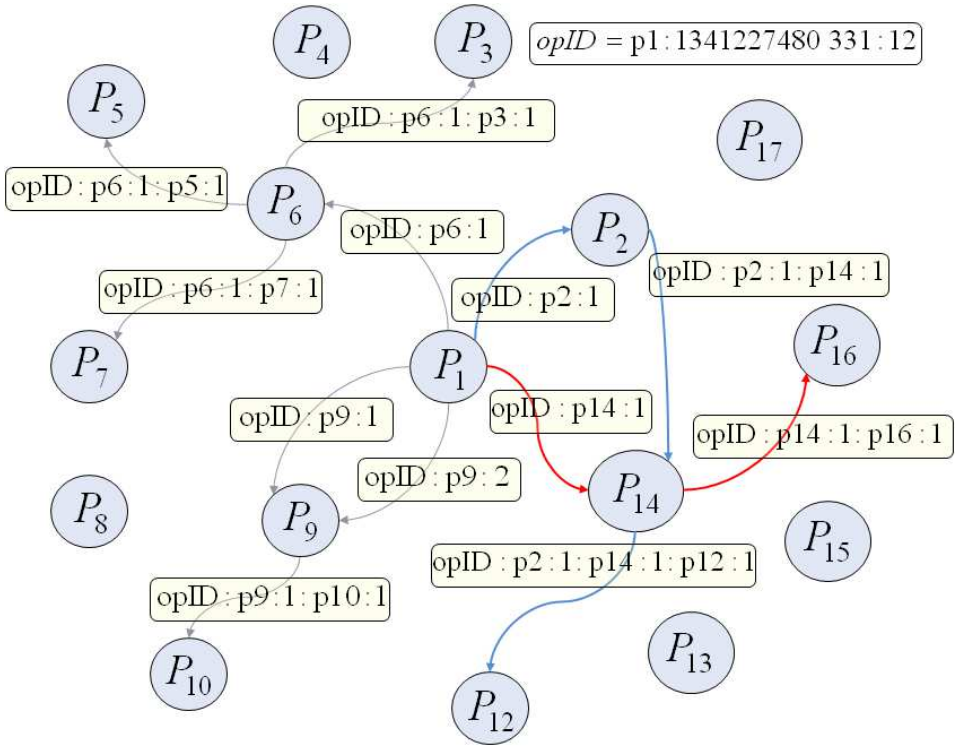


Figure 30 – Cooperation Mesh Constituting an Operation

Figure 30 presents an example of a mesh of inter-peer cooperations resulting from a single operation, illustrating the identifiers of the involved cooperations. That operation (identified as $opID = \text{p1:1341227480331:12}$), is initiated by peer $P_1$. To adequately perform it, $P_1$ solicits cooperations from peers $P_2$, $P_6$, $P_{14}$ and $P_9$. From the latter one it actually requests two different cooperations, in the scope of the same operation. These are therefore identified as $\text{opID:p9:1}$ and $\text{opID:p9:2}$, respectively.

The peers initially contacted by $P_1$ eventually solicit the cooperation of further peers, and so on, and thus, the operational mesh of cooperations expands throughout the system's tissue.

Two of the cooperation chains (sequences of cooperations were each is caused by the previous one, and causes the next one, in the chain), are highlighted with different colours to emphasise a specific possible operational occurrence. That occurrence consists of a peer $P_{14}$, being involved in two different cooperation chains (of the same operation), at different depths of those chains. In the "red" chain $P_{14}$ is directly contacted by $P_1$, and in the "blue" chain, it is reached, by the cooperation mesh of operation $opID = \text{p1:1341227480331:12}$, via $P_2$. $P_{14}$ is thus part of the two following cooperation chains:

- $\text{opID:p14:1} \leftrightarrow \text{opID:p14:1:p16:1}$
- $\text{opID:p2:1} \leftrightarrow \text{opID:p2:1:p14:1} \leftrightarrow \text{opID:p2:1:p14:1:p12:1}$

However, in spite of the possible complexity of the cooperation mesh that results of the execution of an operation, once the above defined scheme of operation and cooperation identification is employed, (for the identification of the exchanged messages), no cross-communication or message switching will occur and the operations will unfold successfully.

### *5.2.3 Peer Identification*

Every system peer has a system-wide unique identifier. It is composed of the following parts:

- the identifier of the system;
- the identifier of the type of peer;
- the peer's unique alphanumeric serial;

Thus, for instance, in a system identified as *p2ptube*, a peripheral peer with serial *xpto001* will be identified *p2ptube:ppeer:xpto001.*

## 5.3 IPCL Operation

### *5.3.1 Introduction*

The IPCL, operates in a purely P2P fashion. It provides the upper layers with communication services, through the exchanging of information packets directly between peers.

### *5.3.2 Interfacing*

IPCL instances interface with the local PLL instance "above" them, and with remote other IPCL instances. As such an IPCL instance exchanges information with both those entities, in both directions and (given the P2P nature of the system), in both the client and server roles.

The purpose of the IPCL is to attend to the requests of the PLL. Doing so, basically consists only of relaying internal outbound PLL messages to external PLL instances (PLL instances of remote peers), and external inbound PLL messages to the local PLL.

To supply such a service, the IPCL requires the delivery of the following parameters:

- from the local PLL instance, (in the case of an outbound internal PLL message):
  - the IP address of the destination peer;
  - the PLL message to be sent;
- from a remote IPCL instance, (in the case of an inbound external PLL message):
  - the actual exchanged IPCL message (from which the PLL message is extracted and passed up to the PLL);

Outbound messages are sent over TCP/IP to the destination peer. Inbound ones are relayed to the local PLL.

## 5.4 PLL Operation

### *5.4.1 Introduction*

The PLL offers a set of core functionalities to enable:

- the security (confidentiality, integrity, authenticity and non-repudiatiability), of all messages exchanged between peers;
- the registration, removal, login and logoff of peripheral peers from the system (in a client/server manner);
- the registration and removal of outer core peers (client/server manner);
- the resolution of peer contact endpoints and authentication info, (in a hybrid manner);
- the shunning and readmission of peripheral and outer core peers (client/server manner).

Every peer has a unique identifier as well as a unique asymmetrical key pair composed of a public key, ($K_{P_i}$), and a private key ($K_{P_i}^{-1}$), which it employs for its authentication. The

ultimate register of the public part of this authentication information, ( $p_i, K_{P_i}$, ), and of the contact endpoints of peers is the $CCP$'s PLL. It is assumed that the PLL of all peripheral and outer core peers "knows" $CCP$'s public key.

The following sections explain, in detail, how the PLL provides the above mentioned functionalities.

### *5.4.2 Secure Messaging*

Inter PLL communication security is assured by the PLL itself. It thus assures the confidentiality, integrity, authenticity and non-repudiatiability of all messages involved in that communication.

The procedure employed to assure communicational security varies slightly depending on whether the communicating peers have already established a communication session or not.

#### *5.4.2.1 Pre-Connection Secure Messaging*

The procedure employed to assure that two peers may communicate securely, before they have established a communication session between them, is always the same regardless of the types of conversing peers and of the role they play in the interaction ("client" or "server").

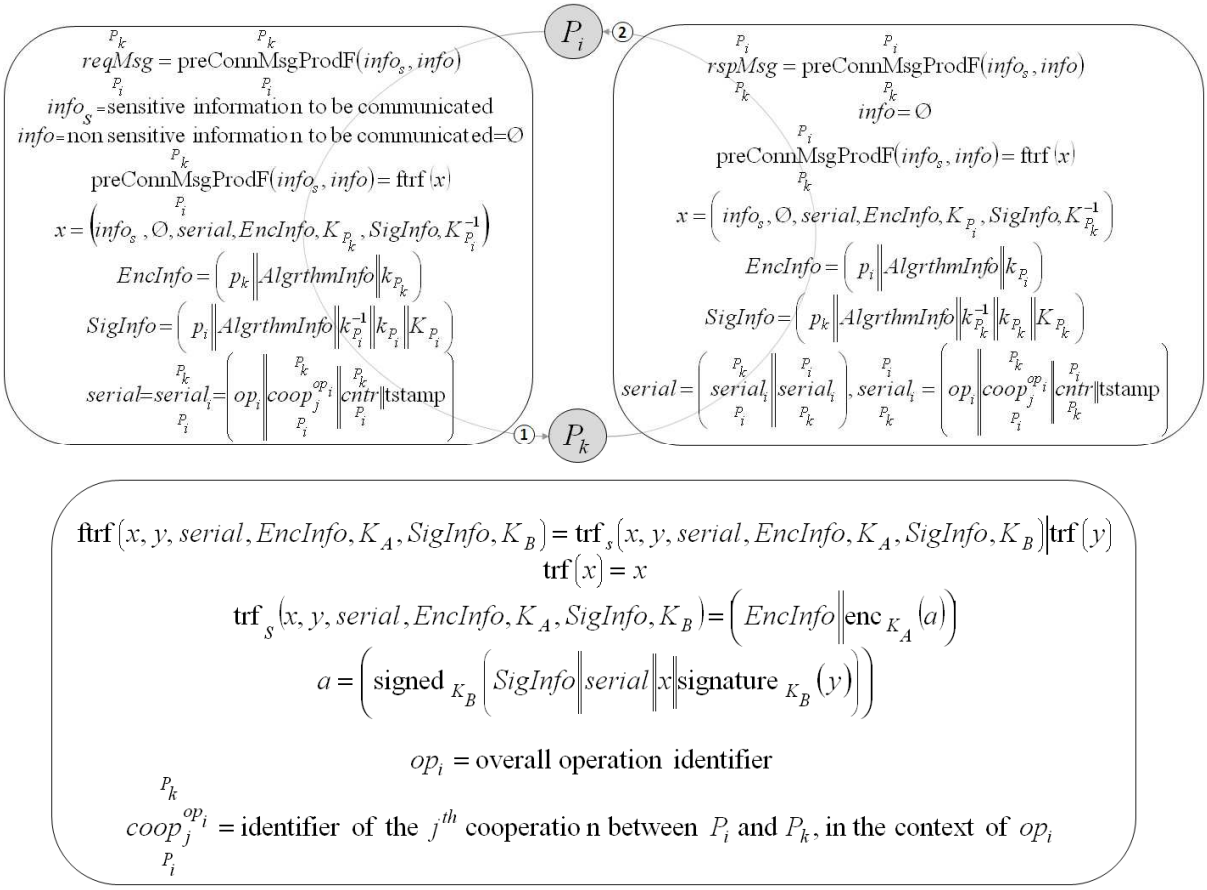


Figure 31 – Pre-Connection Secure Messaging Procedure

Figure 31 exemplifies the procedure in scope (taking into consideration the notation defined in section 5.1 of the present chapter), with generic peer $P_i$ playing the client role and generic peer $P_k$ playing the server role.

The interaction initiating peer, $P_i$, begins by sending the message $reqMsg_{P_i}^{P_k} = \text{preConnMsgProdF}_{P_i}^{P_k}(info_s, info)$ to $P_k$ expressing its request (step 1), where $\text{preConnMsgProdF}_{P_i}^{P_k}(info_s, info) = \text{ftrf}\left(info_S, Ø, serial, EncInfo, K_{P_k}, SigInfo, K_{P_i}^{-1}\right)$. That message contains a transformed (by function $\text{ftrf}(\ldots)$), version of the information to be sent.

That function takes as parameters the following set:

- $info_S$ – is the sensitive information being transmitted. It must always carry, besides other information, the specification of the type the request;
- $info$ – in the non-sensitive information being transmitted, which, in this case, is null;
- $serial$ – is a serial identifier that enables the logical binding between messages and the operations and cooperations that they pertain to (see section 5.2.2). It also performs the unique differentiation between all exchanged messages so that no two messages are ever alike. If a repetition is ever detected, it is either an error or a replay attack and the message must be discarded. The serial carries the following data:
  - the identifier of the overall operation which encompasses the cooperation to which the message pertains;
  - the identifier of the inter-peer cooperation to which the message pertains. Given the way its structure is defined, (see section 5.2.2), this identifier carries the identification of all the previous cooperations in the cooperation chain that lead to the current cooperation. It is the last link in the chain that actually defines the present cooperation. It includes in its structure the identification of the sender and receiver peers;
  - $cntr_{P_i}^{P_k}$ – a cumulative counter of all the messages sent from $P_i$ to $P_k$ in the course of the present cooperation, preceded by the "*req*" or "*resp*" string if the comprising message is coming from the "client" or "server" peer;
  - a timestamp indicating the message's creation time;

  The set composed by the identifier of the inter-peer cooperation and $cntr_{P_i}^{P_k}$ constitutes the comprising message's identifier, in accordance with what was defined in section 5.2.2
- $EncInfo$ - carries the necessary information to enable the authorized receiving party to decode the message. It thus contains: the identification of the peer whose public key was employed to cipher the overall message ($P_k$), so that only such peer may decipher it; the definition of the ciphering algorithm; the identifier of said public key;
- the public key of the destination peer (it is employed to encrypt the message's secure content);
- $SigInfo$ - carries the necessary information to enable the authorized receiving party to validate the signature of message's contents. It thus contains: the identification of

the peer whose private key was employed to sign the overall message ($P_i$); the definition of the signing algorithm; the identifier of the employed private key; the identifier of the corresponding public key; and the actual public key. $SigInfo$, which will be sent to the receiving peer, includes the public key, of the sender peer, in the cases were the peer is performing its initial connection to the *CCP* (registering), as in that case the *CCP* (receiving peer) does not yet know that information. In all other cases, the value of the public key will not be sent as the destination peer must discover it through the adequately reliable means – asking the *CCP* (or some *OCP*);

- the private key of the source peer (it is employed to sign the message's content);

Function $\text{ftrf}\left(x,y,serial,EncInfo,K_A,SigInfo,K_B\right)$, performs its work by concatenating the $SigInfo$ parameter with the serial identifier, with $info_S$ and with the signature (by $P_i$) of $info$, and by signing that set with $P_i$'s private key, and ciphering the result with $P_k$'s public key. The ciphered result is then concatenated with $EncInfo$. The information block resulting from the previous operations is them concatenated with $info$.
The signing process, enables the receiving peer to verify the integrity and origin authenticity of the $\left(SigInfo \| serial \| info_S \| signature_{K_{P_i}^{-1}}(info)\right)$ set and, consequently, of $info$. It also guarantees the non-repudiability, by $P_i$, of that information. The ciphering process ensures the confidentiality of the exchanged information, as only $P_k$ may decipher it. The presence of the $EncInfo$, in plain text, concatenated with the ciphering result is necessary in order to enable the receiving peer, at the PLL, to know how to proceed to unpack it (what keys to employ). The fact that $EncInfo$ is in plain text poses no security problems, as it does not carry any secret key.

In the next step of the interaction (step 2), $P_k$ responds to $P_i$, by sending it the message $\underset{P_k}{\overset{P_i}{rspMsg}} = \text{preConn}\underset{P_k}{\overset{P_i}{\text{Msg}}}\text{ProdF}(info_s, info)$, expressing its response, where $\text{preConn}\underset{P_k}{\overset{P_i}{\text{Msg}}}\text{ProdF} = \text{ftrf}\left(info_s,Ø,serial,EncInfo,K_{P_i},SigInfo,K_{P_k}^{-1}\right)$. The structure of this message, and the way it is produced, is similar to the structure and production mode of $\underset{P_i}{\overset{P_k}{reqMsg}}$. The only differences are the following:

- $info_s$ must, now, always carry, besides other information, the specification of the type of response;
- the employed serial identifier ($serial = \left(\underset{P_i}{\overset{P_k}{serial_j}} \| \underset{P_k}{\overset{P_i}{serial_j}}\right)$), is now composed by two parts. The first one is the serial that was received in the request message. The second part is the actual serial of the $P_k$ -> $P_i$ direction. The latter construct, carries

the same data as the previous with the exception that the counter value may be different and that the timestamp's value is necessarily different;

- the $\left( SigInfo \| serial \| info_S \| signature_{K_{P_k}^{-1}}(info) \right)$ information set now contains the signature of *info* by $P_k$ (instead of $P_i$), and is now signed with $P_k$'s private key and ciphered with $P_i$'s public key.

The structure of the response serial enables:

- the unique identification of the message and thus, its differentiation from all others;
- the logical association, at the receiving end, of the response message to the original request message to which its responds;
- the proving, by $P_k$, that it, in fact possesses $P_k$'s private key ($K_{P_k}^{-1}$), and thus is $P_k$-

  This is so because the $\left( SigInfo \| serial \| info_S \| signature_{K_{P_k}^{-1}}(info) \right)$ information set is signed with $K_{P_k}^{-1}$. That set contains (inside *serial*) the serial of the request message. This serial was never produced before, and thus, its inclusion, in the response serial, and signing, proves that the response message cannot be a repetition of any kind and that $P_k$ has therefore just now produced it, employing $K_{P_k}^{-1}$.

Obviously, the interaction between $P_k$ and $P_i$, may continue with further requests from $P_i$ and responses from $P_k$. The employed messages would have the same structure, and production mode as explained above, but, from this point (step 2) on, the *serial* in the request messages would also include the new part of the *serial* of the previous response messages in the interaction. Thus, for instance, in a hypothetical next request $j+1$, (from $P_i$ to $P_k$), which is following up on the previous response $j$ (from $P_k$ to $P_i$), we would have that $serial = \left( {}^{P_i}_{P_k}serial_j \| {}^{P_k}_{P_i}serial_{j+1} \right)$.

This chaining of serials enables that which was described in the previous bullets.

The procedure above described, (pre-connection secure messaging), is only employed when a peer is in the process of registering (see section 5.4.3 of the present chapter), with the system (communication with *CCP*), or in the (post-registering) process of establishing a communication session (see section 5.4.5 of the present chapter) with another peer.

### *5.4.2.2 Post-Connection Secure Messaging*

The procedure employed to assure that two peers may communicate securely, after they have established a communication session between themselves, is always the same regardless of the types of conversing peers and of the role ("client" or "server"), that they play in the interaction.

Figure 32 exemplifies said procedure (taking into consideration the notation defined in section 5.1 of the present chapter), with generic peer $P_i$ playing the client role and generic peer $P_k$ playing the server role.

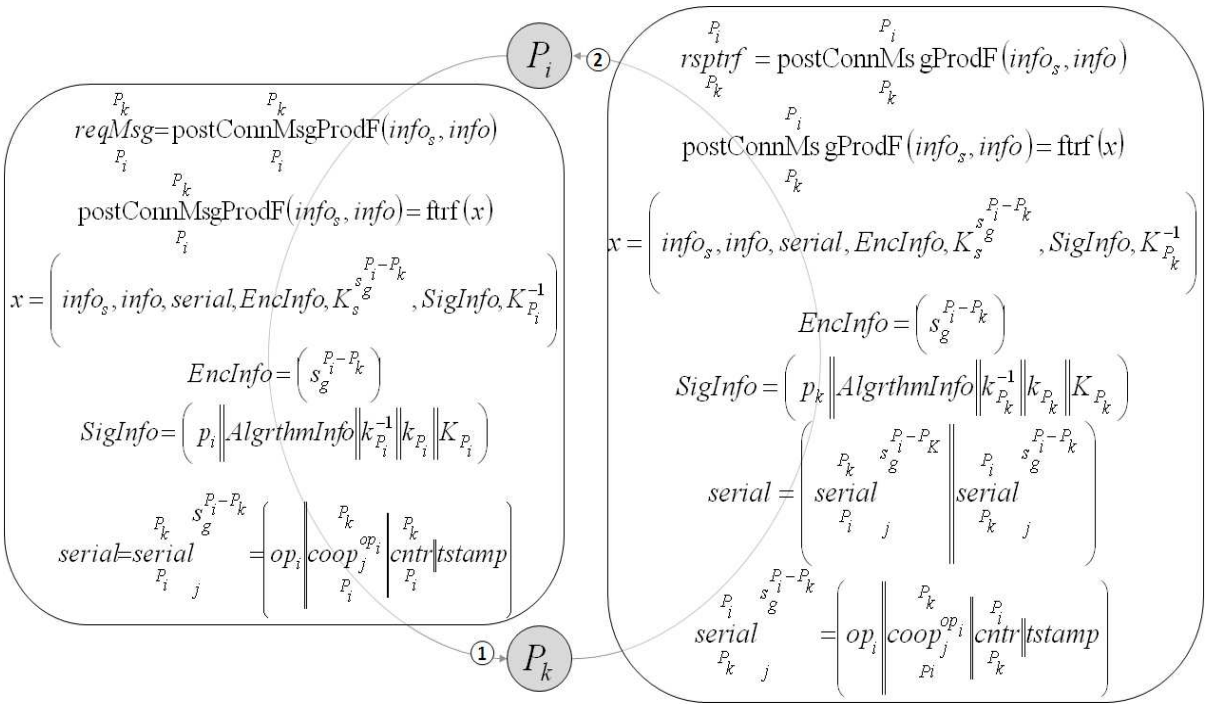


Figure 32 – Post-Connection Secure Messaging Procedure

In step 1, $P_i$ sends a $\overset{P_k}{\underset{P_i}{reqMsg}} = \text{postConnMsgProdF}\left(info_s, info\right)$ message to $P_k$ expressing its request, where $\overset{P_i}{\underset{P_k}{\text{postConnMsgProdF}}} = \text{ftrf}\left( info_s, info, serial, EncInfo, K_s^{P_i - P_k}, SigInfo, K_{P_k}^{-1} \right)$.

That message contains a transformed version of the information to be sent (achieved by function $\text{ftrf}\left(.....\right)$). That function takes as parameters the following set:

- $info_s$ – is the sensitive part of the information being transmitted. It must always carry, besides other information, the specification the type of the request;
- $info$ – is the non-sensitive part of the information being transmitted;
- $serial$ – is a serial identifier that enables the unique differentiation between all exchanged messages so that no two messages are ever alike. If a repetition is ever detected, it is either an error or a replay attack and the message must be discarded. The serial carries the following data:
  - the identifier of the overall operation which encompasses the cooperation to which the message pertains;
  - the identifier of the inter-peer cooperation to which the message pertains. Given the way its structure is defined, (see section 5.2.2), this identifier carries the identification of all the previous cooperations in the cooperation chain that lead to the current cooperation. It is the last link in the chain that actually defines the present cooperation. It includes in its structure the identification of the sender and receiver peers;

- o $cntr_{P_i}^{P_k}$ – a cumulative counter of all the messages sent from $P_i$ to $P_k$ in the course of the present cooperation, preceded by the "*req*" or "*resp*" string if the comprising message is coming from the "client" or "server" peer;
- o a timestamp indicating the message's creation time;

The set composed by the identifier of the inter-peer cooperation and $cntr_{P_i}^{P_k}$ constitutes the comprising message's identifier, in accordance with what was defined in section 5.2.2

- $EncInfo$ - carries the necessary information to enable the authorized receiving party to decode the message. It carries only $s_g^{P_i-P_k}$ (the identifier of the communication session, between $P_i$ and $P_k$, in which takes place the inter peer cooperation that the message belongs to), as that information suffices for the receiving peer to be able to contextualise the message and know how to decode it;
- the secret session key $K_{s_g^{P_i-P_k}}$, (it is employed in the message's ciphering);
- $SigInfo$ - carries the necessary information to enable the authorized receiving party to validate the signature of message's contents. It thus contains: the identification of the peer whose private key was employed to sign the overall message ($P_i$); the definition of the signing algorithm; the identifier of the employed private key; the identifier of the corresponding public key; and the actual public key;
- the private key of the source peer;

Function $\text{ftrf}\left(x,y,serial,EncInfo,K_A,SigInfo,K_B\right)$, performs its work by concatenating $SigInfo$ with the serial identifier, with $info_s$, and with the signature (by $P_i$) of $info$, and by signing that set with $P_i$'s private key, and ciphering the result with the secret session key, $K_{s_g^{P_i-P_k}}$. It then concatenates the ciphering result with $EncInfo$. The information block resulting from the previous operations is them concatenated with $info$.

The signing processes enables the receiving peer to verify the integrity and origin authenticity of the $\left(SigInfo \| serial \| info_S \| signature_{K_{P_i}^{-1}}(info)\right)$ set and, consequently, of $info$. It also guarantees the non-repudiability, by $P_k$, of that information. The ciphering process ensures the confidentiality of the sensitive exchanged information, $\left(SigInfo \| serial \| info_S \| signature_{K_{P_i}^{-1}}(info)\right)$, as only $P_k$ knows $K_{s_g^{P_i-P_k}}$. The presence of the communication session's identifier, in plain text (inside $EncInfo$), concatenated with the ciphering result is necessary in order to enable the receiving peer, at the PLL, to know the message's context, and thus, how to proceed to unpack it (what keys to employ).

In the next step of the interaction (step 2), $P_k$ responds to $P_i$, by sending it the message $\underset{P_k}{rspMsg}^{P_i} = \text{postConn}\underset{P_k}{\text{MsgProdF}}^{P_i}(info_s, info)$, expressing its response, where $\text{postConn}\underset{P_k}{\text{MsgProdF}}^{P_i} = \text{ftrf}\left(info_s, info, serial, EncInfo, K_s^{s_g^{P_i-P_k}}, SigInfo, K_{P_k}^{-1}\right)$. The structure of this message, and the way it is produced, is similar to the structure and production mode of $\underset{P_i}{reqMsg}^{P_k}$. The only differences are the following:

- $info_s$, must, now, always carry, besides other information, the specification the type of the response;
- the employed serial identifier ($serial = \left( \underset{P_i \quad j}{serial}^{\,s_g^{P_i-P_K}_{P_k}} \,\middle\|\, \underset{P_k \quad j}{serial}^{\,s_g^{P_i-P_k}_{P_i}} \right)$), is now composed by two parts. The first one is the serial that was received in the request message. The second part is the actual serial of the $P_k$ -> $P_i$ direction. The latter construct, carries the same data as the previous with the exception that the counter value may be different and that the timestamp's value is mandatorily different;
- the $\left( SigInfo \,\|\, serial \,\|\, info_s \,\|\, signature_{K_{P_k}^{-1}}(info) \right)$ information set is now signed with $P_k$'s, and not $P_i$'s, private key.

The structure of the response serial enables:

- the unique and authenticated identification of the message and thus, its differentiation from all others and its binding to its comprising operation and cooperation;
- the logical association, at the receiving end, of the response message to the request message to which its responds;

Obviously, the interaction between $P_k$ and $P_i$, may continue with further requests from $P_i$ and responses from $P_k$. The employed messages would have the same structure and production mode as explained above, but, from this point on (step 2), the $serial$ in the request messages would also include the new part of the $serial$ of the previous response message in the interaction. Thus, for instance, in a hypothetical next request $j+1$, (from $P_i$ to $P_k$), which is following up on the previous response $j$ (from $P_k$ to $P_i$), we would have that $serial = \left( \underset{P_i \quad j}{serial}^{\,s_g^{P_i-P_K}_{P_k}} \,\middle\|\, \underset{P_k \quad j+1}{serial}^{\,s_g^{P_i-P_k}_{P_i}} \right)$.

This chaining of serials enables that which was described in the previous bullets.

### *5.4.3 Peer Registration*

Before a peripheral or outer core peer is ready to start its participation in the system, it must first go through an initialization or registration phase (that takes place at the PLL).

If at a given moment, (such as the moment of their original start-up), a specific peer knew nothing about any other system peers, it would be impossible for it to ever become connected to the system. To prevent such situations a bootstrapping procedure is employed: all system peers have hardcoded into them, the unchanging IP address of the *CCP* and its authentication information (public key).

#### *5.4.3.1 Peripheral Peer Registration*

This process is performed in a client/server manner, (between the *PP* and the *CCP* respectively), the first time a peripheral peer interacts with the system. The process, occurring at the PPL, is depicted in Figure 33 (taking into consideration the notation defined in section 5.1 of the present chapter).

The registering peripheral peer, $PP_i$, begins by sending $CCP$ the message $\underset{PP_i}{\overset{CCP}{regreqmsg}} = \text{preConn}\underset{PP_i}{\overset{CCP}{\text{Msg}}}\text{ProdF}(info_s, info)$, where $info_s = (\text{regreq} \| sid_{PP_i}), info = \text{Ø}$ (step 1).

In accordance with what was specified in section 5.4.2.1, $\underset{PP_i}{\overset{CCP}{regreqmsg}}$ will contain the information block $\left( SigInfo \| serial \| \text{regreq} \| sid_{PP_i} \right)$, signed with $PP_i$'s private key (assuring its integrity and authenticity of origin), and encrypted with the $CCP$'s public key ($K_{CCP}$), so that only $CCP$ may read it (assuring confidentiality).

$regreq$ is the identifier of the type of message. $sid_{PP_i}$ is a signed package ($\text{signed}_{K^{-1}_{PP_i}}\left( id_{PP_i} \right)$) containing $PP_i$'s identification information ($id_{PP_i} = \left( pp_i \| AlgrthmInfo \| k_{PP_i} \| k^{-1}_{PP_i} \| K_{PP_i} \right)$). It is signed by $PP_i$ so that any system peer can assert that $PP_i$ validates that information (its thus cannot be faked, even by $CCP$). It is signed independently from (and redundantly to) the overall message so that it may latter be independently redistributed.

$CCP$ then verifies the validity of the received message. If all is correct, it sends back (step 2) a challenge message to $PP_i$, to check if it truly possesses the private key corresponding to $K_{PP_i}$, (that $PP_i$ specified in $sid_{PP_i}$), as the message received in step 1, even if signed with said private key, may be a repetition attack. In the challenge message $info_s = regchlg, info = \text{Ø}$. Said message, (in accordance with the process described in section 5.4.2.1), contains the $(SigInfo \| serial \| \text{regchlg})$ set. $SigInfo$ carries the necessary information to enable the validation of message content's signature (mentioned next in the text), $\text{regchlg}$ is the identification of the type of message. $serial$ carries a composed serial, consisting of a concatenation of the serial received in the request message, plus a new

serial $\underset{CCP}{serial}_j^{PP_i}$. This content is signed by $CCP$, and is encrypted with $K_{PP_i}$ so that only $PP_i$ may read it (assuring confidentiality). By containing the older serial it proves that $CCP$ is who it claims to be, as it has signed a random unique peace of data, generated and delivered by $PP_i$. $PP_i$ will fully prove its possession of $K_{PP_i}^{-1}$ by signing the newer ($CCP$ generated) serial (which, in accordance with what was specified in section 5.4.2.1, it has to do).
$PP_i$ thus proceeds to produce a third serial, concatenate it with the one received from $CCP$, and sign the set of serials, which it then sends to $CCP$ (step 3).

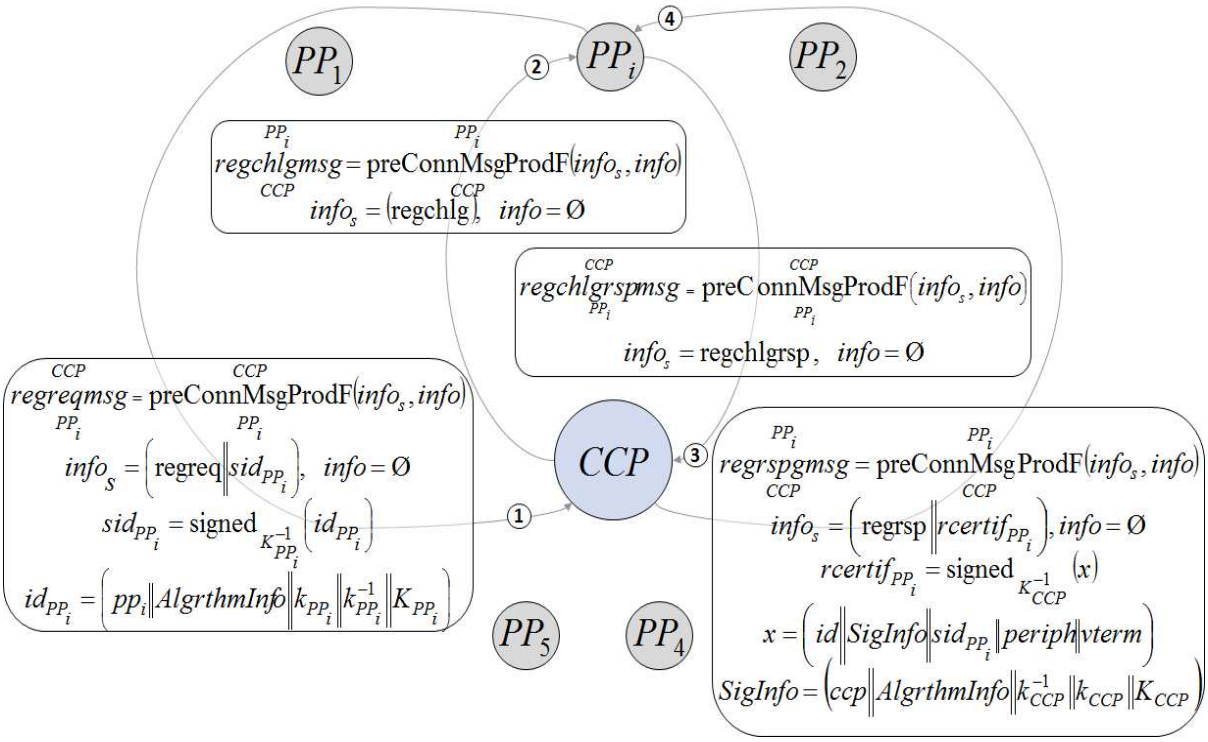


Figure 33 – Peripheral Peer Registration Process

Finally $CCP$ validates the signature of the received serials. If all is ok, at this point, both $PP_i$ and $CCP$ are assured that their partner, in the interaction, is in fact in possession of the private key it claims to be, and thus, is who it claims to be. As such, $CCP$ produces $PP_i$'s registration certificate, ($rcertif_{PP_i} = \text{signed}_{K_{CCP}^{-1}}(id \| SigInfo \| sid_{PP_i} \| periph \| vterm)$), and sends it to $PP_i$ (step 4). This certificate contains its identifier ($id$), $SigInfo$ (carries the necessary information to enable the validation of the certificate's signature, where $SigInfo = (ccp \| AlgrthmInfo \| k_{CCP}^{-1} \| k_{CCP} \| K_{CCP})$), $sid_{PP_i}$, the indication of the role that $PP_i$ plays in the system (peripheral peer), and the validity term after which it expires. It is signed independently from (and redundantly to) the overall message so that it may latter be independently redistributed. It may be freely exchanged amongst the peer community as a proof of $PP_i$'s and $CCP$'s mutual acceptance of $PP_i$'s participation in the system, (as a peripheral peer), and of its public authentication credentials.

### 5.4.3.2 Outer Core Peer Registration

The initialization of an outer core peer consists of its registration with $CCP$, its retrieval, from $CCP$ of the system's data structure and of the definition of its interval of servicing responsibility, over the peripheral peer collective.

The registration procedure of outer core peers is in all similar to the homologous procedure for peripheral peers (explained in 5.4.3.1 of the present chapter). The main differences are the following:

- given the sensitive role that $OCPs$ play in the system, they are not to be simple user owned peers. Instead, just as is the case for the $CCP$, $OCPs$ are especially powerful peers owned and operated by the system's operating entity. This way the acceptance, by $CCP$, of a specific $OCP_i$, does not depend only on $OCP_i$ proving its possession of its claimed original key pair. Whenever an $OCP$ adhesion request is received by the $CCP$, the latter will require user (system operating entity), input accepting the $OCP$'s adhesion. Thus, an $OCP$'s registration is a user assisted process;
- the registering outer core peer, $OCP_i$, is given a registration certificate, $rcertif_{OCP_i}$, which validates its role as an outer core peer, instead of as a peripheral peer.

After an $OCP$ registers, it will then open a communication session with $CCP$ (in the manner explained in section 5.4.5), which will remain opened indefinitely. Once the above defined process is completed the $CCP$ will automatically, instruct the $OCP$ , to store the system's data structure (in the manned described in section 5.4.8.6) and also instruct it about its interval of servicing responsibility (in the manner described in section 5.4.8.7).

### 5.4.3.3 Post Registration Actions

The registration of a new peer means that the system's data model needs to be updated to account for said peer. The $CCP$ updates its own copy (the main copy), of said structure, as it is processing the request of the prospective new peer (if the request is accepted). It later proceeds to the updating of the data structures, at all $CCPs$, in the manner described in section 5.4.8.6 of the present chapter.

## 5.4.4 Peer Registration Update

Once a peer's registration certificate expires or if, for some reason, a peer's key pair is compromised, the peer must update its registration. To do so (considering that the peer already has a communication session established with the *CCP*), it issues a registration update request (in the secure manner explained in section 5.4.2.2), where $info_s = \left( \text{regupdreq} \| sid_{PP_i} \right)$, and $sid_{PP_i}$ contains the peer's new identification parameters. If all is ok the *CCP* responds (in the secure manner explained in section 5.4.2.2), with the peer's new registration and connection certificates.

If there is not even the possibility of securely establishing a communication session with the *CCP*, then, the registration update process will be in all similar to the one described in section 5.4.3, with the exception that $info_s = \left( \text{regupdreq} \| sid_{PP_i} \right)$ and not $info_s = \left( \text{regreq} \| sid_{PP_i} \right)$.

### *5.4.5 Communication Session Establishment*

Before any two peers can communicate they must first establish a communication session between them. This is true for inter-peripheral, periphery to outer core, peripheral to central core and intra-core peer communication.

This PLL process enables the combination, between the communicating peers, of a secret symmetric key, ($K_s^{s_g^{P_i-CCP}}$), to be employed in the ciphering of all later communication between them, within that session (the $g^{th}$ session). Taking into consideration the notation defined in section 5.1 (of the present chapter), this process, occurring at the PPL, is depicted in Figure 34, for the connection between a generic peer, $P_i$ and $CCP$.

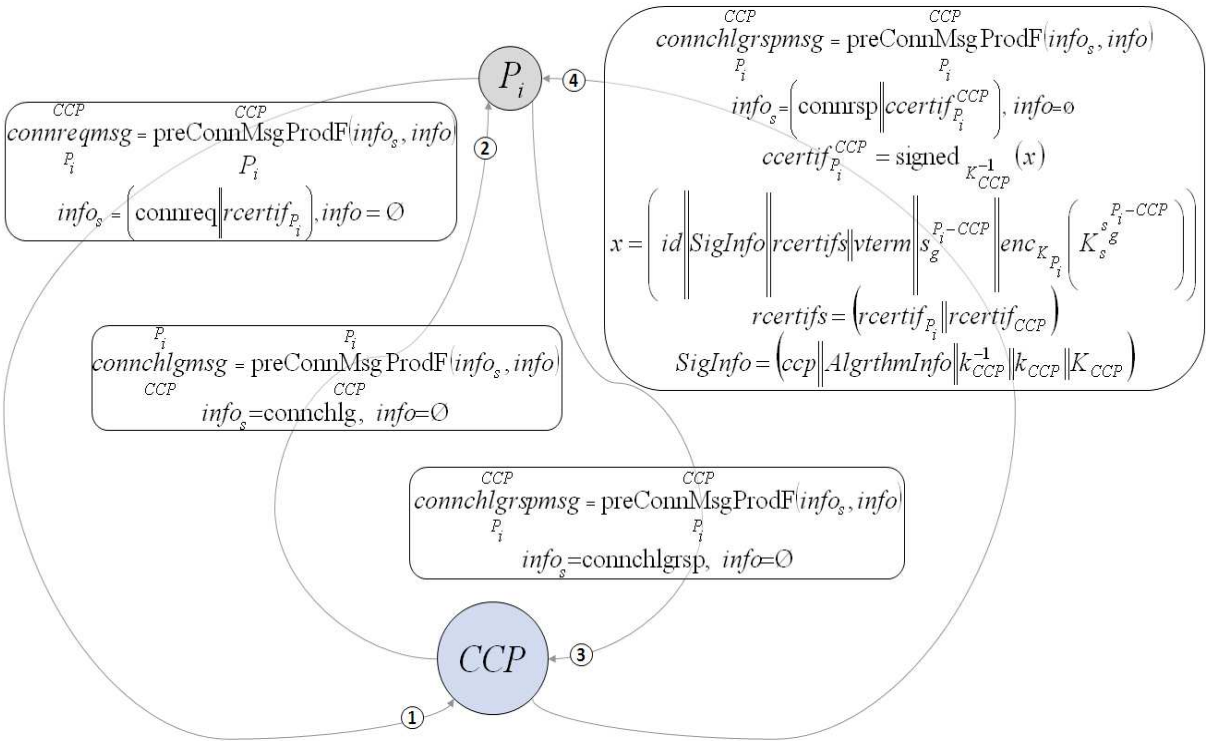


Figure 34 – Inter-Peer Communication Session Establishment Process

This processes' initial step (step 1), is similar to the first step described in the registration process (section 5.4.3.1 of the present chapter). The difference being that the sent message contains the registration certificate ($rcertif_{PP_i}$) of the communication initiating peer ($P_i$) instead of its $sid_{P_i}$.

$CCP$ obtains the identification information for $P_i$ from its database, and validates the message and $rcertif_{P_i}$ (if the "server" peer, in this interaction was not $CCP$ but a peripheral peer, it would obtain the requesting peer's identification information in the manner explained in section 5.4.7 of the present chapter). If all is correct, it then proceeds to step 2, sending a challenge message to $P_i$, to check the veracity of its claimed identity. That message, in accordance with what is defined in section 5.4.2.1, contains its $SigInfo$ field, the identifier of the type of message ($\text{connchlg}$) and the serial sent by $P_i$, as well as a second new

unique serial generated by $CCP$, that $P_i$ will have to sign to fully prove its identity. These contents of the challenge message are then signed by $CCP$, (which proves that $CCP$ in fact possesses $K_{CCP}^{-1}$, and thus is $CCP$), encrypted with $K_{P_i}$ so that only $P_i$ may decipher it (assuring confidentiality), and concatenated with $EncInfo$.
In a third step, $P_i$ proceeds (in accordance with the procedure defined in section 5.4.2.1) to sign the $CCP$ generated serial, plus a third new serial (plus $\mathrm{connchlgrsp}$), and sends the result back to $CCP$ (step 3).
$CCP$ then validates the signature of the serial it sent to $P_i$ in step 2. If all is correct $CCP$ produces the session identifier ($s_g^{P_i-CCP}$), and the secret session encoding key ($K_s^{s_g^{P_i-CCP}}$). Employing those two objects, plus $rcertif_{PP_i}$, $rcertif_{CCP}$, the adequate $SigInfo$ object, and a validity term, $CCP$ builds the connection certificate $ccertif_{P_i}^{CCP}$ (with a unique $id$ of its own), and sends it to $P_i$ (step 4). $ccertif_{P_i}^{CCP}$ is signed by $CCP$, enabling the verification of its validity, throughout the system, in a distributed manner. From this point on, the communication session between the two peers is considered open. Communication, between the two peers will proceed in accordance with what is defined in section 5.4.2.2.

In the case where the peer playing the server role is the $CCP$, (as in the provided example), this procedure is equivalent to the login of the client peer with the system, and the connection certificate may also be seen as a login certificate.

The above presented description, exemplifies the establishing of a communication session, between a generic peer and the $CCP$. The actual specific types of communication session establishments that will occur in the system (in terms of the involved peers), and their relevant details are the following:

- the establishment, by peripheral peers, of communication sessions with:
  - the $CCP$:
    - for a peripheral peer to be eligible to participate in the system it must have a opened communication session with $CCP$ (the equivalent of login into the system). Thus, whenever a registered peripheral peer, $PP_i$, comes on-line, it must first establish a new communication session with $CCP$, which should remain open until $PP_i$ goes offline;
    - in step 4 the sent communication certificate will also include the identity of the $OCP$ which will be responsible for handling the "read-only" requests of the peripheral peer, until the end of the session. Thus

$$ccertif_{PP_i}^{CCP} = signed_{K_{CCP}^{-1}}\left( id \left\| SigInfo \right\| rcertifs \left\| vterm \right\| s_g^{PP_i-CCP} \left\| ocp_x \right\| enc_{K_{P_i}}\left( K_s^{s_g^{PP_i-CCP}} \right) \right)$$

  - outer core peers:
    - in the step corresponding to step 1, (from Figure 34), the sent

certificate will be the connection certificate ($ccertif_{PP_i}^{CCP}$) that $PP_i$, obtained when it connected/logged in with the $CCP$;

- other peripheral peers;
  - in the step corresponding to step 1, (from Figure 34), the sent certificate will be the connection certificate ($ccertif_{PP_i}^{CCP}$) that $PP_i$, obtained when it connected/logged in with the $CCP$;

- the establishment, by outer core peers, of communication sessions with:
  - the $CCP$:
    - for an outer core peer to be eligible to participate in the system it must have a opened communication session with $CCP$ (the equivalent of logging into the system). Peripheral peers will be intermittently connected to the system, and thus frequently establish new communication sessions with the peers that they need to communicate to. Thus, communication sessions established between peripheral peers and $CCP$ will typically be of a relatively short duration. Oppositely, outer core peers are to be permanently available. The first time an outer core peer, $OCP_i$, comes on-line, (once it has registered with $CCP$), it will establish a communication session, with $CCP$, that will only end when $OCP_i$ ceases to operate, or if the session key is compromised and needs to be replaced;
  - other outer core peers;
    - in the step corresponding to step 1, (from Figure 34), the sent certificate will be the connection certificate ($ccertif_{PP_i}^{CCP}$) that $OCP_i$, obtained when it registered with the $CCP$.

### *5.4.6 Peer Login*

Once a peer has registered with the system, every time it comes online, before it can actually begin interacting with it, said peer must first login. That process basically consists of the establishment of a communication session with the $CCP$, in the manner explained in section 5.4.5 of the present chapter. In the course of the process the peer, that is logging in, is given a connection certificate. Said certificate may, in this case be seen, also, as a login certificate.

If the connecting peer is peripheral, the responsibility for servicing its login request must be attributed to some specific $OCP$. This is done through the procedure outlined in section 5.4.8.7 of the present chapter.

The login of a peer means that a new entity is now available to cooperate with the system. This is a relevant piece of information that needs to be stored in the system's data structure, so that it may be coherently handled, accessed and updated. The $CCP$ updates its own copy of that structure, while processing the login request (if it is accepted). It later proceeds to the updating of the data structures, at all $OCPs$, in the manner described in section 5.4.8.6 of the present chapter.

### *5.4.7 PLL Info Discovery*

The regular operation of any peer, at the PLL, implies that peripheral peers will frequently need to discover information on other peers (such as identifier, IP address, public key, whether it is quarantined or expelled, etc.), so that they can communicate in a reliable manner.

The process, undertaken by peripheral peers, to obtain any PLL information is performed in a client/server or hybrid manner. This PLL process happens as follows (taking into consideration the notation defined in section 5.1 of the present chapter).

Assuming that a communication session has already been established between the inquiring peripheral peer (e.g. $PP_i$), and its servicing $OCP$ , the earlier begins by sending (in the secure manner defined in section 5.4.2.2 of the present chapter), to the latter, a request message ($\underset{PP_i}{\overset{OCP_j}{\text{reqpllinfo}}}\left(info_S, info\right)$), where the sensitive part of the information, $info_S = \left(\text{reqpllinfo} \| infoDef\right)$, contains a parameter, ($\text{reqpllinfo}$), which identifies the message as a request for some PLL info, and a second parameter ($infoDef$ ), indicating the specific information that it wants.
$OCP_j$ may immediately send back a message, with the required data ($\underset{OCP_j}{\overset{PP_i}{rsppllinfo}}\left(info_S, info\right)$ , $info_S = \left(\text{rsppllinfo} \| \text{the desired data}\right)$), or it may send back a message redirecting $PP_i$ to some other peripheral peer $PP_j$ (or outer core peer), where $info_S = \left(\text{rsppllinfo} \| retrpermit_{pllInfo}^{PP_i} \| pllildo_s\right)$. That redirection message, contains, besides the message type identifier, an, *OCP* signed, PLL info retrieval permit ($retrpermit_{pllInfo}^{PP_i}$), enabling it to retrieve the desired information, and the $pllildo_s$ object, which describes the location(s) from where $PP_i$ may retrieve said information. The retrieval permit may be described as

$$retrpermit_{pllInfo}^{PP_i} = \text{signed}_{K_{OCP_j}^{-1}}\left(\text{id} \| \text{pllInfoRetrPerm} \| SigInfo \| pp_i \| vterm \| infoDef \| id_{pllildo_s}\right).$$ It contains the permit's identifier, its type identifier, the permit's adequate $SigInfo$ (where $SigInfo = \left(ocp_j \| AlgrthmInfo \| k_{OCP_j}^{-1} \| k_{OCP_j} \| K_{OCP_j}\right)$), the identifier of the peer to which the permit is granted, the permit's validity term, the definition of the specific information which may be retrieved ($infoDef$ ) and the identifier of the $pllildo_s$ object.

As stated, $infoDef$ , is the part of the original request, and of the permit, that identifies the type of information that is desired or which may be retrieved, respectively. It may bear the following values:

- if the desired information is the identification (location and authentication), information on some specific peer (e.g. $PP_k$), then $infoDef = \left(pIDInfo \| pp_k\right)$. In this case, the information object that will eventually be obtained is

$ppInfo_s = \text{signed}_{K^{-1}_{OCP_j}}\left(id \| SigInfo \| rcertif_{PP_k} \| IP_k \| vterm\right)$, where *vterm* is this object's validity term date;

- if the desired information is the quarantine or expulsion lists then $infoDef = (quarList)$ or $infoDef = (expList)$ respectively. In this case, the information object that will eventually be obtained, will respectively be $quarList_s = \text{signed}_{K^{-1}_{OCP_j}}(SigInfo \| QList \| tstamp)$, or $expList_s = \text{signed}_{K^{-1}_{OCP_j}}(SigInfo \| XList \| tstamp)$, where *tstamp* indicates the moment of creation of those objects, so that the peers employing them may know how recent that information is.

In both such cases $SigInfo = \left(ocp_j \| AlgrthmInfo \| k^{-1}_{OCP_j} \| k_{OCP_j} \| K_{OCP_j}\right)$.

The PLL Information Location Describing Object may be defined as $pllildo_s = \text{signed}_{K^{-1}_{OCP_j}}\left(id \| SigInfo \| infoDef \| vterm \| \left(rcertif_{PP_i} \| ..... \| rcertif_{PP_n}\right)\right)$. It carries $SigInfo$ (which takes the same form as the one that was just described above), the definition of the information set to who's location it pertains ($infoDef$), its validity term and a list of the registration certificates of the peers from where the information, defined in $infoDef$, may be retrieved.

The signing, by $OCP_j$, of the retrieval permit and of $pllildo_s$, even if redundant (as the sensitive part of the PLL message is signed by the sending peer), is necessary to enable the latter independent reuse of those objects and their communication to other peers in an authenticable form. Both those objects carry inside the identification of their issuing peers, for much the same reason, so that any peer may authenticate said objects.

If $OCP_j$ does redirect $PP_i$, in the next step of the operation, the latter will then establish a communication session with one (or more), of the peers identified in $pllildo_s$ (by their registration certificates), and obtain from it, the desired information. To do so, in its request for the desired information, $PP_i$ will need to send (amongst other fields), both the retrieval permit and $pllildo_s$, so that the contacted peer(s) may validate $PP_i$'s right to retrieve the information in scope.

As can be seen, from the above description, for a peer to be able to communicate with some other peer, it will need some information about it, which it, generally, can only obtain through communication with other peers. If at any moment a peer knew nothing about any other peer it would be stuck in a situation where it would never be able to communicate with any other peer. This is avoided because every peer "knows", from the start, the IP and public key (and its identifier) of the $CCP$ .This way, even if a peer knows nothing else, it may always contact $CCP$ and obtain all the information it needs. Furthermore, if, for some reason, the

necessary $OCP_j$, for obtaining the identification of a target peer, is not available, the request for that information is redirected to $CCP$ or some other $OCP$.

### 5.4.8 Management

#### 5.4.8.1 Introduction

The proper operation of the PLL demands that some management procedures are undertaken periodically, or in response to some event, to assure the overall integrity and cooperativeness of the peer tissue.

These operations are centrally performed by the $CCP$, (based on notifications received form the periphery or outer core), which does all the necessary decision making, and translate into the issuing of $CCP$ instructions to $OCPs$ or $PPs$.

Said notifications may be generally described as $notif^{pll}_{P_i} = \text{signed}_{K^{-1}_{P_i}}\left(SigInfo \| notiftype \| notifcontent\right)$, where $SigInfo$ carries the necessary information to enable the validation of the notification, $notiftype$ designates the type of notification, and $notifcontent$ is the actual content of the notification.
The mentioned instructions may be generally described as $instruct^{pll}_{CCP}{}^{P_i} = \text{signed}_{K^{-1}_{CCP}}\left(SigInfo \| instructtype \| instructcontent\right)$, where $SigInfo$ carries the necessary information to enable the validation of the instruction, $instructtype$ designates the type of instruction, and $instructcontent$ is the actual content of the instruction.

The main PLL management operations can basically be summed up as the redistribution of $PP$ servicing workload of $OCPs$, the neutralization of infringing or faulty behaviour on the part of peers, and the updating of $OCP$ data structures

#### 5.4.8.2 OCP Workload Redistribution

As peripheral peers log into the system, the $CCP$ distributes the responsibility, to service such peers, in an even manner, through the $OCPs$. However, some $PPs$ may represent higher servicing burdens than others. This means that the $CCP$ must somehow obtain information on the work burden of $OCPs$, and proceed to redistribute their servicing assignments, if necessary.

For this reason, every $OCP$ periodically sends (in the secure manner defined in section 5.4.2.2), to $CCP$ an activity notification which may be described as $notifactiv^{pll}_{OCP_i}{}^{T=t} = \text{signed}_{K^{-1}_{OCP_i}}\left(SigInfo \| \text{activity} \| \left(id \| t \| ocp_i \| wrkld \| wrklist\right)\right)$ (in a message where $info_s = \left(\text{notifload} \| notifactiv^{pll}_{OCP_i}{}^{T=t}\right)$). This notification contains the adequate $SigInfo$ object, the definition of the type of notification, its unique identifier, the definition of the time instant which it purports to, the identification of the sending $OCP$, its total work burden (in terms of the percentage of CPU usage), and a list discriminating the division of that burden over the

peripheral peers that It is servicing $wrklist = \left( \left( pp_1 \| wrkld_{PP_1} \right) \| ... \| \left( pp_n \| wrkld_{PP_n} \right) \right)$.

$CCP$ acknowledges the reception of the notification with a PLL messages where $info_s = (\text{notifloadack})$.

The notification object is independently (and redundantly) signed so as to enable their independent treatment and aggregation in an authenticable form.

The $CCP$ collects the mentioned notifications, from all $OCPs$ and, periodically, evaluates the overall situation. If the global workload distribution is uneven, the $CCP$ determines which $PPs$ must be reassigned to which other $OCPs$ and proceeds to instruct those reassignments in the manner described in section 5.4.8.7 of the present chapter.

### *5.4.8.3 Infringing or Faulty Peer Behaviour Neutralization*

The system is composed by a distributed mesh of independently controlled peers. Some are under the control of the system's operating entity and others are controlled by regular users of varied levels of trustworthiness.

There is thus a non-negligible possibility of failure and of infringing behaviour occurring amongst such peers, that the system must be prepared to deal with.

This means that the $CCP$ must somehow obtain information about undesirable behaviour on the part of peers. For this to occur it must be the $PPs$ and $OCPs$ to notify, to $CCP$, whenever some other peer behaves inappropriately towards them. Such reports must carry proof of the claimed misbehaviour.

For the present time the system supports only the reporting of a misbehaviour where the misbehaving peer responds with signed but corrupted or forged information, such as, for instance, a corrupted version of a PLL info retrieval permit (see section 5.4.7).

Thus whenever a peer $P_i$ receives a request or reply message, signed by another peer $P_k$, whose contents are undecipherable or (if it is the case that they are to be signed, for instance, by the $CCP$ or an $OCP$), incorrectly signed, it proceeds to send (in the secure manner defined in section 5.4.2.2), a misbehaviour notification (pertaining to $P_k$) to $CCP$, which may be described as:

$notifmissbehav_{P_i}^{CCP\ pll} = \text{signed}_{K_{P_i}^{-1}}\left( SigInfo \| \text{missbehav} \| \left( id \| t \| p_i \| p_k \| proofObj \right) \right)$, where $p_i$ is the identifier of the sending peer and $proofObj$ is the received signed message. The notification is sent in a PLL message where $info_s = \left( \text{notifbehav} \left\| notifmissbehav_{P_i}^{CCP\ pll} \right. \right)$, which the $CCP$ acknowledges with a response PLL message where $info_s = (\text{notifbehavack})$.

$notifmissbehav_{P_i}^{CCP\ pll}$ is independently (and redundantly to the securing process described in section 5.4.2.2), signed to enable a more independent, and still authenticable, manipulation of it.

The $CCP$ periodically analyses all pending reports and decides what to do regarding the "criticized" peers. It may do nothing, it may suspend or it may expel the peer. In the two latter

cases the $CCP$ proceeds in accordance with what is described in section 5.4.8.4 of the present chapter.

On the other hand, the case of previously shunned peers is also periodically revisited. In such moments $CCP$ may decide to let the peer remain shunned, or re-enable its participation in the system. In the latter case it proceeds in accordance with what is described in section 5.4.8.5.

### *5.4.8.4 Peer Shunning*

The system's peers may, for whatever reason, engage in abnormal, uncooperative or sabotaging behaviour or be frequently employed (by users) for dishonest activities. In such cases, once the $CCP$ detects such patterns (in the manner described in the previous section), it proceeds to shun the infringing, or malfunctioning, peer from the system either in a temporary or permanent base, and to place in a quarantine or expulsion list, respectively. This applies to both peripheral and outer core peers.

#### *5.4.8.4.1 OCP Shunning*

To shun a specific $OCP$ (e.g. $OCP_x$), the $CCP$ proceeds (in the secure manner defined in section 5.4.2.2 of the present chapter) to:

- instruct all $OCPs$ to place $OCP_x$ in the peer quarantine or expulsion list if $OCP_x$ was suspended or permanently banned, respectively. Said instruction may be described as $instrpshunning_{CCP}^{pll\ OCP_i} = \text{signed}_{K_{CCP}^{-1}}\left(SigInfo \| \text{pshunning} \| \left(Q \cup \text{X} \| ocp_x\right)\right)$, where the mentioned $OCP_i$ is the instructed $OCP$. They are carried in PLL messages where $info_s = \left(\text{instrsh} \| instrpshunning_{CCP}^{pll\ OCP_i}\right)$, which are acknowledged, by the destination peers, by sending back a PLL message where $info_s = (\text{instrshack})$. From that point on, no requests from $OCP_x$ will be attended, until further notice;
- reattribute every peripheral peer under the care of $OCP_x$ to other $OCPs$ and inform all relevant peripheral and outer core peers of servicing responsibility reattributions, (in the manner presented in section 5.4.8.7). From that point onward, the affected peripheral peers can no longer resort to the services of $OCP_x$, until further notice.

If it is possible to contact the shunned peer, it will be sent, to that peer as well, the $instrpshunning_{CCP}^{pll\ OCP_i}$ instruction as a manner of informing it that it has been shunned or expelled.

#### *5.4.8.4.2 PP Shunning*

To shun a specific $PP$ (e.g. $PP_x$), the $CCP$ proceeds (in the secure manner defined in section 5.4.2.2 of the present chapter) to:

- instruct all $OCPs$ to place $PP_x$ in the peer quarantine or expulsion list if $PP_x$ was suspended or permanently banned, respectively. Said instruction may be defined as

$instrpshunning_{CCP}^{pll\ OCP_i} = \text{signed}_{K_{CCP}^{-1}}\left(SigInfo \| \text{pshunning} \| \left(Q \text{ or } X \| pp_x\right)\right)$, where the mentioned (in the equation), $OCP_i$ is the instructed $OCP$. They are carried in PLL messages where $info_s = \left(\text{instrsh} \left\| instrpshunning_{CCP}^{pll\ OCP_i}\right.\right)$, which are acknowledged, by the destination peers, by sending back a PLL message where $info_s = (\text{instrshack})$. From that point on, no requests from $PP_x$ will be attended until further notice (including by the $OCP$, which was up until that time responsible for attending its requests).

If it is possible, the shunned peer itself ($PP_x$) will be sent the $instrpshunning_{CCP}^{pll\ PP_x}$ instruction as a manner of informing it that it has been shunned or expelled.

### 5.4.8.5 Peer Readmission

To readmit a previously shunned peer (e.g. $P_x$), the $CCP$ does the following:

- in the secure manner defined in section 5.4.2.2 of the present chapter, the $CCP$ instruct all $OCPs$ to remove the readmitted peer, from the peer quarantine. The structure and contents of the readmission instruction may be described as $instrpreadm_{CCP}^{pll\ OCP_i} = \text{signed}_{K_{CCP}^{-1}}\left(SigInfo \| \text{instrpreadm} \| \left(Q \text{ or } X \| p_x\right)\right)$, where the mentioned $OCP_i$ (in the equation), is the instructed $OCP$. They are carried in PLL messages where $info_s = \left(\text{instrreadm} \left\| instrpreadm_{CCP}^{pll\ OCP_i}\right.\right)$, which are acknowledged, by the destination peers, by sending back a PLL message where $info_s = (\text{instrshack})$. From that point on, requests from $P_x$ will again be attended;
- If it is readmitting an $OCP$ (e.g. $OCP_x$):
  - $CCP$ then performs a peripheral peer servicing workload reassignment, so as to attribute to the readmitted peer a set of peripheral peers for it to service;
- If it is readmitting a $PP$ (e.g. $PP_x$):
  - $CCP$ then assigns some $OCP$ to service $PP_x$'s requests and informs the earlier of that reassignment;
- If the readmitted peer ($P_x$) remained online:
  - it is sent the $instrpreadm_{CCP}^{pll\ P_x}$ instruction as a manner of informing it that it has been readmitted.
- else:
  - when the peer attempts to re-establish a communication session with the $CCP$ (i.e. loggin), it will be allowed to do so.

### 5.4.8.6 OCP Updating

Whenever some change or addition needs to be made to the system's data structure (at the PLL level), it is the $CCP$ that coordinates that activity. The $CCP$ performs such changes on

the version of the data structure which it locally stores, but it then becomes necessary to update all the other versions of the data structure, stored at the $OCPs$. Such an updating/initialization of an $OCP$'s data structure is also necessary when an $OCP$ first registers and connects to the system.

To perform such an updating/initialization of the $OCP$ data structures, the $CCP$ issues instructions, to the relevant set of $OCPs$, (from one to all of them), commanding them to perform the necessary update. Said instructions may be defied as, $instrupdatedb_{CCP}^{pll\ OCP_i} = \text{signed}_{K_{CCP}^{-1}}\left(SigInfo \| \text{instrupdatedb} \| id \| updInfo\right)$ (for an exemplifying $OCP_i$ target), where, $updInfo$ is a set of one or more SQL commands and respective data parameters. They are carried in PLL messages where $info_s = \left( \text{instrupd} \| instrupdatedb_{CCP}^{pll\ OCP_i} \right)$ and $info$ is an archive of whatever data objects need to be transferred. $OPCs$ proceed to do as they are instructed and, after that task is performed, they send back a success PLL message, to the $CCP$, where $info_s = (\text{instrupdack})$.

If the updating of any of the $OPCs$ fails, the $CCP$ retries the operation, until it succeeds, up to a specific maximum amount of attempts. If it proves impossible to update a specific $OCP$, the $CCP$ proceeds to disconnect it from the system in the manner explained in section 5.4.8.4.1.

### *5.4.8.7 PP Servicing Assignment*

The assignment of peripheral peer servicing responsibilities is done by the $CCP$. It consists of assigning the responsibility to service the requests of a specific $PP$ to a specific $OCP$.

The $CCP$ makes an assignment decision and then it informs the relevant affected peers, by sending, to which ever of them is necessary, the, appropriate instruction. That may be defined as $instrpserviceassign_{CCP}^{pll\ OCP_i} = \text{signed}_{K_{CCP}^{-1}}\left( SigInfo \| \text{pserviceassign} \| ccertif_{P_i}^{CCP} \right)$. It contains a fresh, $CCP$ generated, login certificate (see section 5.4.6), which contains the identity of the serviced $PP$ and of its handler $OCP$. That instruction is carried in a PLL message where $info_s = \left( \text{instrpserv} \| instrpserviceassign_{CCP}^{pll\ OCP_i} \right)$. The targeted peers respond with a PLL message where $info_s = (\text{instrpservack})$.

### *5.4.9 UEL Info Relaying*

The main purpose of the PLL, besides the servicing of inter-PLL requests, is to attend to the requests of the UEL. Doing so basically consists of relaying internal outbound UEL messages to external UEL instances (UEL instances of remote peers), and external inbound UEL messages to the local UEL.

The outbound UEL messages are wrapped in PLL messages (see section 7.3.1 of the present chapter). This implies the signing (with the local peer's private key) and ciphering (with the secret session key), of the earlier (together with some other content), their

wrapping in the latter message and the latter message's delivery to the IPCL together with the IP address of the destination peer (the communication port could simply be a constant and universally established one).

The inbound PLL messages are deciphered and their signature is validated. If all is ok, the UEL message is extracted and (if necessary), reconstituted, and then relayed to the local UEL, together with the validated identifier of the message's sending peer (at the PLL).
All of the above-mentioned PLL messages have a structure where $info_s = \left(\text{uelmrelay} \| UELInfo_S\right)$.

## 5.5 UEL Operation

### *5.5.1 Introduction*

The UEL handles, in a client/server manner all the operations related to:

- the registration, login, logoff and removal of users;
- the maintenance of user accounts and performing of currency transactions;
- the injection, removal, updating and versioning of MOs;
- the injection and removal of MO Ransom Announcements (see section 7.4.8);
- the collection of information on user actions.

In all such cases, the $CCP$ 's UEL plays the server role and the involved $PP$'s UEL, plays the client role. The clients will typically send their requests to the server side, which judges their legitimacy and, if adequate, enforces them or provides some required information, if that is the case.

A mixed P2P and client/server operation mode is employed, by UEL, in:

- the semantics based searching for MOs (and MO Ransom Announcements);
- the retrieval/distribution of MOs (and MO Ransom Announcements);
- the searching and retrieval/distribution of information objects pertaining to users or other aspects of the system.

The *CCP*'s UEL instance holds a permanently updated copy of the system's UEL global data structure. Furthermore, as explained in section 3.4.3, that information is also replicated over each of the outer core peers.

Whenever some fragment of that information is needed, peripheral peers will request it from their handling $OCP$, which may respond by handing it over directly or by informing the "client", about the appropriate "server" $PP(s)$, for the acquisition of the desired information.

### *5.5.2 Secure Messaging*

#### *5.5.2.1 Introduction*

The division between sensitive and non-sensitive information, which exists in PLL messages, translates into a similar division in UEL messages. These, therefore, have a sensitive ($UELinfo_s$), and a non-sensitive part ($UELinfo$). The earlier part will be carried in the sensitive part of PLL messages, and the latter part will be carried in the non-sensitive part of PLL messages.

The PLL's regular operation guaranties the confidentiality and integrity of communications in the P2PTube architecture. Thus, in terms of its internal operations, the UEL does not need to perform any further information obfuscation procedures.

Users, however, are external and independent entities from the system, which have their own authentication credentials and whose involvement needs to be validated, in order for their requests to be fulfilled. For this reason, whenever an UEL interaction is caused by the attending (at a peer $PP_i$), of a user request, the emitted UEL message(s), must bare proof of the user's involvement.

### *5.5.2.2 Pre User Login Secure Messaging*

A peer will only engage in UEL cooperations before a user, hosted on it, logs into the system in the following situations (or without any user hosted on it):

- it is registering a user to the system;
- its is login a user into the system;
- it is participating as a "server" in a UEL cooperation, where its services were requested by some remote peer.

The procedure employed to securely exchange UEL messages in the situations were a user is involved is depicted in Figure 35 (taking into consideration the notation defined in section 5.1 of the present chapter). The purpose of said procedure is to provide proof of the involved user's (if any) involvement in the UEL cooperation.

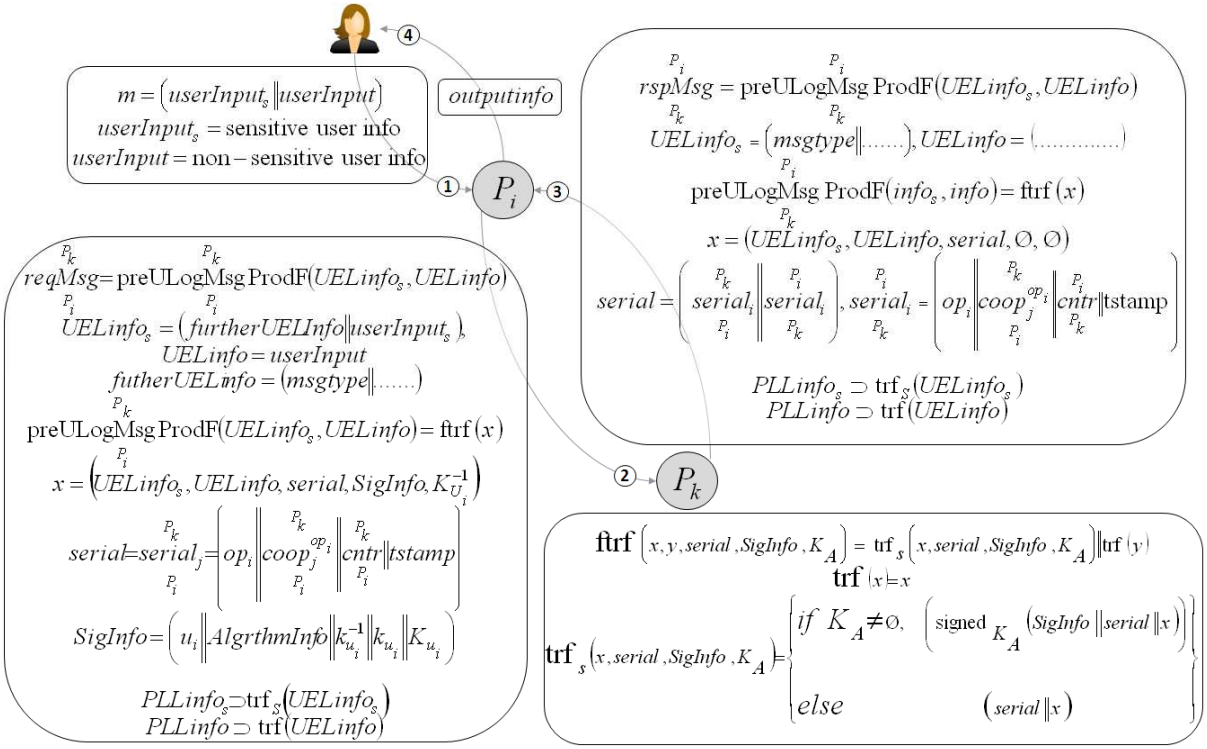


Figure 35 – Pre User Login Secure Messaging Procedure

In said figure, a generic peer $P_i$, is hosting user $U_i$ and, thus, playing the client role. The server role is plaid by generic peer $P_k$ (in reality in pre-user login UEL cooperations involving users it will typically be the $CCP$ to play the server role).

At an initial moment $U_i$ issues some specific request to his hosting peer $P_i$. In doing so $U_i$ may deliver both sensitive and non-sensitive information. $P_i$ performs whatever initial processing is necessary of the received information and builds, $UELinfo_s$ and $UELinfo$. The

earlier carries whatever sensitive information needs to transmitted (received from the user or otherwise). This information will always include the designation of the message's type. The latter carries the non-sensitive information.

$P_i$ then builds the UEL message to be sent. Said message may be defined as

$$\underset{P_i}{\overset{P_k}{reqMsg}} = \text{preULog}\underset{P_i}{\overset{P_k}{\text{Msg}}}\text{ProdF}(UELinfo_s, UELinfo) = \text{ftrf}\left(UELinfo_s, UELinfo, serial, SigInfo, K_{U_i}^{-1}\right).$$

The execution of function $\text{ftrf}\left(x, y, serial, SigInfo, K_A\right)$, over the given arguments results in the production of two information blocks: $\text{signed}_{K_{U_i}^{-1}}\left(SigInfo \| serial \| UELinfo_s\right)$; and $UELinfo$.

The UEL message is logically composed of the two above mentioned information blocks.

The $serial$ is an UEL level serial identifier. Such an independent serial (from the PLL level serial mentioned in earlier sections), is necessary to perform the unique identification of UEL messages. This identification enables the logical binding of those messages to the UEL operations and cooperations that they pertain to (see section 5.2.2). It also performs the unique differentiation between all exchanged UEL messages so that no two messages are ever alike. This uniqueness assurance is necessary also at the UEL level, in spite of the same already existing for the PLL messages, because UEL communication errors may occur, even between "honest peers", and UEL messages be duplicated. If a repetition is ever detected, it is either an error or a replay attack and the message must be discarded. The serial carries the following data:

- the identifier of the overall operation which encompasses the cooperation to which the message pertains;
- the identifier of the inter-peer UEL cooperation to which the message pertains. Given the way its structure is defined, (see section 5.2.2), this identifier carries the identification of all the previous UEL cooperations in the cooperation chain that lead to the current cooperation. It is the last link in the chain that actually defines the present UEL cooperation. It includes in its structure the identification of the sender and receiver peers;
- $\underset{P_i}{\overset{P_k}{cntr}}$ – a cumulative counter of all the messages sent from $P_i$ to $P_k$ in the course of the present cooperation, preceded by the "*req*" or "*resp*" string if the comprising message is coming from the "client" or "server" peer;
- a timestamp indicating the message's creation time;

The process of signing $\left(SigInfo \| serial \| UELinfo_s\right)$, when it occurs, enables the receiving peer to verify that the integrity of that information set and serves as proof of $U_i$'s involvement. It thus guarantees the non-repudiability, by $U_i$, of that information.

Once the message is built, $P_i$ sends it, (step 1), to $P_k$. To do so, the UEL message is passed, as single object to the local PLL. At the PLL, however, the UEL message is split, $UELinfo_s$ is carried in the sensitive part of the PLL message and $UELinfo$ in the non-sensitive part of that message.

In the next step of the cooperation (step 2), $P_k$ responds to $P_i$, by sending it the message $\underset{P_k}{\overset{P_i}{rspMsg}} = \text{preULog}\underset{P_k}{\overset{P_i}{\text{Msg}}}\text{ProdF}(info_s, info) = \text{ftrf}\left(UELinfo_s, UELinfo, serial, \emptyset, \emptyset\right)$, expressing

its response. The structure of this message, and the way it is produced, is similar to the structure and production mode of ${}^{P_k}_{P_i}reqMsg$. The only differences are the following:

- $info_s$ must, now, always carry, besides other information, the specification the type of the response instead of the specific type of the request;
- the employed serial identifier ($serial = \left( {}^{P_k}_{P_i}serial_j \middle\| {}^{P_i}_{P_k}serial_j \right)$), is now composed by two parts. The first one is the serial that was received in the request message. The second part is the actual serial of the $P_k$->$P_i$ direction. The latter construct, carries the same data as the previous with the exception that the counter value may be different and that the timestamp's value is mandatorily different;
- the $\left( SigInfo \| serial \| UELinfo_s \right)$ information set is now only $\left( serial \| UELinfo_s \right)$ and it is no longer subjected to any signature, as there is no user on the responding side whose presence needs to be proven.

The structure of the response serial enables:

- the unique identification of the message and thus, its differentiation from all others;
- the logical association, at the receiving end, of the response message to the original request message to which its responds;

Obviously, the interaction between $P_k$ and $P_i$, may continue with further requests from $P_i$ and responses from $P_k$. The employed messages would have the same structure, and production mode as explained above, but, from this point (step 2) on, the $serial$ in the request messages would also include the new part of the $serial$ of the previous response message in the interaction. Thus, for instance, in a hypothetical next request $j+1$, (from $P_i$ to $P_k$), which is following up on the previous response $j$ (from $P_k$ to $P_i$), we would have that $serial = \left( {}^{P_i}_{P_k}serial_j \middle\| {}^{P_k}_{P_i}serial_{j+1} \right)$.

This enables that which was described in the previous bullets.

As it was implicitly shown above, the $UELinfo_s$ parts of UEL messages which are originated as a response to UEL requests from other peers (response message from $P_k$ to $P_i$ in Figure 35), will be signed only by the emitting peer's private key. Given that the PLL already performs such a signing, it will not actually be done at the UEL, but at the PLL instead.
The same will happen with the $UELinfo_s$ parts of UEL request messages that do not involve any user, such as UEL messages containing commands (from the $CCP$).

The only exceptions to the above situations occur when the UEL information is to be later super-distributed. In such cases it will be redundantly signed, at the UEL, with the peer's private key.

#### 5.5.2.3 Post User Login Secure Messaging

The overall procedure employed by the system to ensure messaging reliability, at the UEL, after a user has logged into the system, hosted at some peer, is practically the same as the one described in section 5.4.2.1 (pre user login secure messaging). The existing differences, which occur at the client side (when hosting a user and sending its request), are the following:

- the user's hosting certificate, $hostcertif_{U_i}^{PP_i}$, (see section 5.5.5), is an added parameter that is supplied to $\text{ftrf}(....)$, together with $SigInfo$. The combined presence of this certificate, and the signing of the $\left(SigInfo \| hostcertif_{U_i}^{PP_i} \| serial \| UELinfo_s\right)$ parameter set, proves the user's involvement dispensing a challenge-response procedure.

### 5.5.3 User Registration

This (pre user login) UEL process is performed in a client/server manner and is depicted (taking into consideration the notation defined in section 5.1), in Figure 36.
The registering user, ($U_i$), delivers, to her hosting peer, ($PP_i$), her public identification package, $id_{U_i}$, (containing $u_i$, $uname_{U_i}$, the identifiers of the user's public and private keys, and the actual public key itself $K_{U_i}$), and also her private key, $K_{U_i}^{-1}$ (step 1).

$PP_i$ produces $sid_{U_i}$, (by concatenating the signing algorithm describing information with $id_{U_i}$, and signing the set with $K_{U_i}^{-1}$), and concatenates it with the identification of the type of request ($\text{uregreq}$). It thus builds $UELinfo_s$. $PP_i$ then performs $\text{ftrf}(....)$ on that set (in accordance with what was defined in section 5.5.2.2), and builds a UEL message, which it sends it to $CCP$ (step 2). The signature of $id_{U_i}$ (which results in the production of $sid_{U_i}$), with $K_{U_i}^{-1}$, validates the origin of $sid_{U_i}$, impedes any later repudiation of $sid_{U_i}$ (by $U_i$), and enables a later, independent, distribution of that data object. The information in scope is then sent, within the sensitive part of the PLL message (in the secure manner defined in section 5.4.2.2), thus assuring its confidentiality, integrity and origin authenticity.

$CCP$ then verifies the validity of the received message. If all is ok, the process continues.
Given the upstream security level, provided by the PLL, and the signing, with the user's private key, of the received message (which carries inside unique serial identifiers), one could think that the reception of this message (when correctly built), by the *CCP*, should be enough for the latter peer to accept the registration of $U_i$. However, it may be that the PLL security provisions of $PP_i$ were compromised, and an attacker knows its private key (but not that of $U_i$), and is intercepting its communications. Such an attacker may be replaying some captured UEL messages to cause disruption in the system. It is for this reason that challenge and challenge response steps are necessary, i.e. to verify the validity of the specified user credentials.

In light of the above, $CCP$ sends back (step 3) a challenge message to $PP_i$ in order to check if it truly possesses the private key of $U_i$.

In the challenge message $UELinfo_s = (\text{uregchlg}), UELinfo = Ø$. Said message, (in accordance with the process described in section 5.5.2.2), contains the $(SigInfo \| serial \| \text{uregchlg})$ set. uregchlg is the identification of the type of message. $serial$ carries a composed serial, consisting of a concatenation of the serial received in the request message, plus a new serial $\underset{CCP}{\overset{PP_i}{serial_j}}$. $PP_i$ will have to sign the newer ($CCP$ generated), UEL level, serial, (of which it could have no prior knowledge), with $U_i$'s private key to prove that it truly knows it.
$PP_i$ proceeds, in accordance to what was defined in section 5.5.2.2, to produce a third serial, concatenate it with the one received from $CCP$, and sign the set of serials, which it then sends to $CCP$ (step 4).

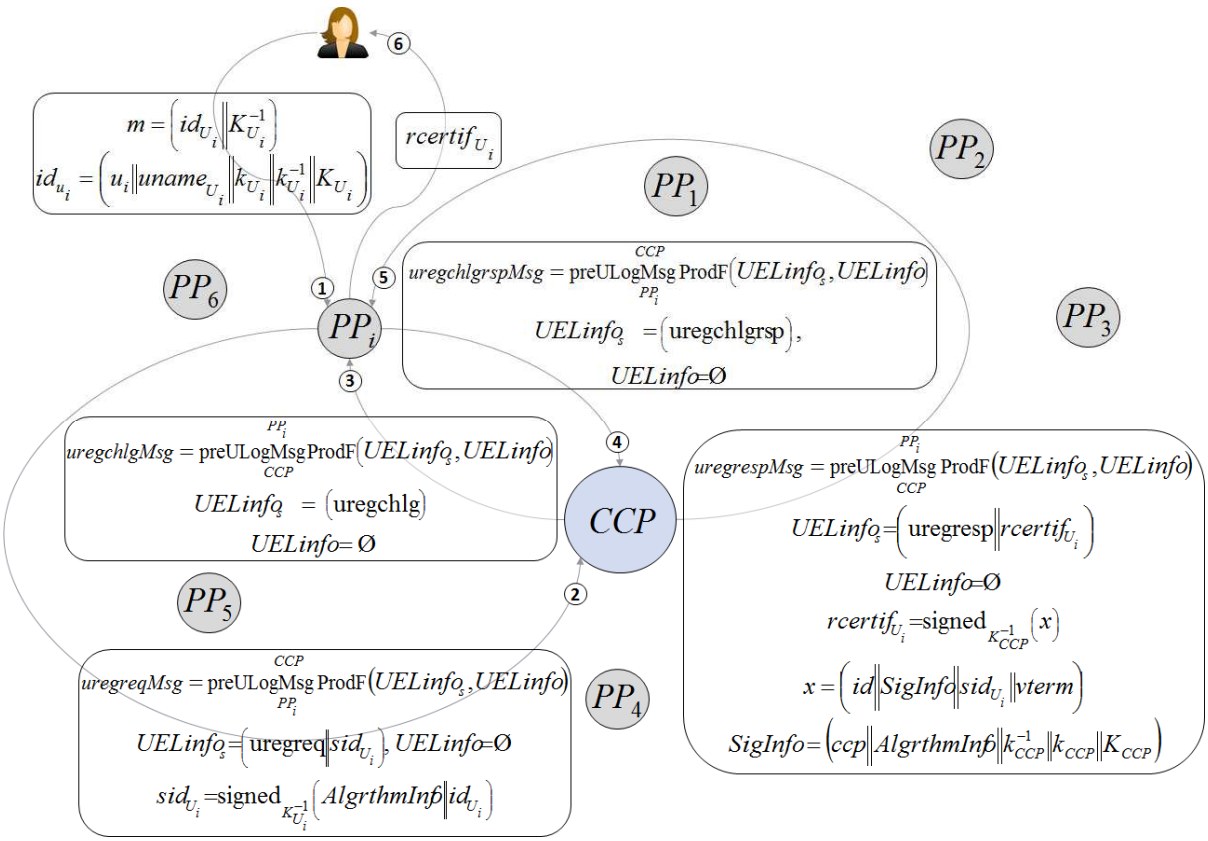


Figure 36 – User Registration Procedure

Finally $CCP$ validates the signature of the received serials. If all is ok $CCP$ registers the new user, and produces a user registration certificate ($rcertif_{U_i} = \text{signed}_{K_{CCP}^{-1}}(id \| SigInfo \| sid_{U_i} \| vterm)$), validating the user's participation in the system, which it sends (step 5), to $PP_i$, included in the sensitive part of a PLL message. The certificate is signed by the $CCP$ and contains, its identifier, $SigInfo$ (where $SigInfo = (ccp \| AlgrthmInfo \| k_{CCP}^{-1} \| k_{CCP} \| K_{CCP})$), $sid_{U_i}$ and the validity term after which it expires, (and a new one must be obtained by the user).

Finally $PP_i$ delivers $rcertif_{U_i}$ to $U_i$.

Once the user registration is complete, the $CCP$ takes care of sending the relevant parts of the information, newly added to the system's data structure, to all the $OCPs$ (in the manner presented in section 5.5.7.7), so that the latter may update their copy of the data structure.

In the user registration process, the user's public key is being delivered, for the first time, to the $CCP$, and the corresponding private key employed for the first time. Thus a man-in-the-middle attack (from an attacker that, somehow, knows $PP_i$'s private key but not $U_i$'s), may succeed in sending wrong information to the $CCP$. However the registration certificate produced by the $CCP$ will contain those wrong values, and, when passed back to the original peer, this will detect the error and the operation will fail.

### *5.5.4 User Registration Update*

Once a user's registration certificate expires or if, for some reason, a user's key pair is compromised, said user must update its registration. To do so (considering that the user is already logged into the system), it issues a registration update request (in the secure manner explained in section 5.5.2.3), where $UELinfo_s = (\text{uregupdreq} \| sid_{U_i})$, and $sid_{U_i}$ contains the user's new identification parameters. If all is ok the *CCP* responds (in the secure manner explained in section 5.5.2.3), with the user's new registration and hosting certificates.

If there is not even the possibility of securely logging the user into the system, then, the registration update process will be in all similar to the one described in section 5.5.3, with the exception that $UELInfo_s = (\text{uregupdreq} \| sid_{U_i})$ and not $UELInfo_s = (\text{uregreq} \| sid_{U_i})$.

### *5.5.5 User Login*

A user authentication enables the user to enjoy of all the system's services. This (pre user login), UEL process is performed in a client/server manner and is depicted (taking into consideration the notation defined in section 5.1), in Figure 37.

$U_i$ delivers her authentication data to her hosting peer, $PP_i$ (step 1). That package contains her registration certificate, and private key.
$PP_i$ communicates with $CCP$ (step 2), indicating that it wishes to login $U_i$. The employed message has $UELinfo_s = (\text{uauthreq} \| rcertif_{U_i})$, where $\text{uauthreq}$ is the designation of the type of request, and $rcertif_{U_i}$ provides the full identification of the user that the peer wants to login. $CCP$ responds (step 3), with a challenge message, whose (UEL) serial $PP_i$ will have to sign (in the course of producing the challenge response message), with $U_i$'s private key, to prove that it is in fact hosting $U_i$.
$PP_i$ then returns to $CCP$, a signed package (see section 5.5.2.2) containing the designation of the message type ($\text{uauthchlgrsp}$), as well as the mentioned serial, thus proving that it is indeed hosting $U_i$ (step 4).

If all is correct, $CCP$ then returns a user hosting certificate (step 5), which may be described as $hostcertif_{U_i}^{PP_i} = \text{signed}_{K_{CCP}^{-1}}\left(id \| SigInfo \| rcertif_{U_i} \| pp_i \| vterm\right)$. It is signed by $CCP$ and contains $rcertif_{U_i}$ (for the clear identification of the hosted user), its identifier, $pp_i$ (for the identification of the hosting peer), and a validity term after which the certificate expires and a new one must be obtained.

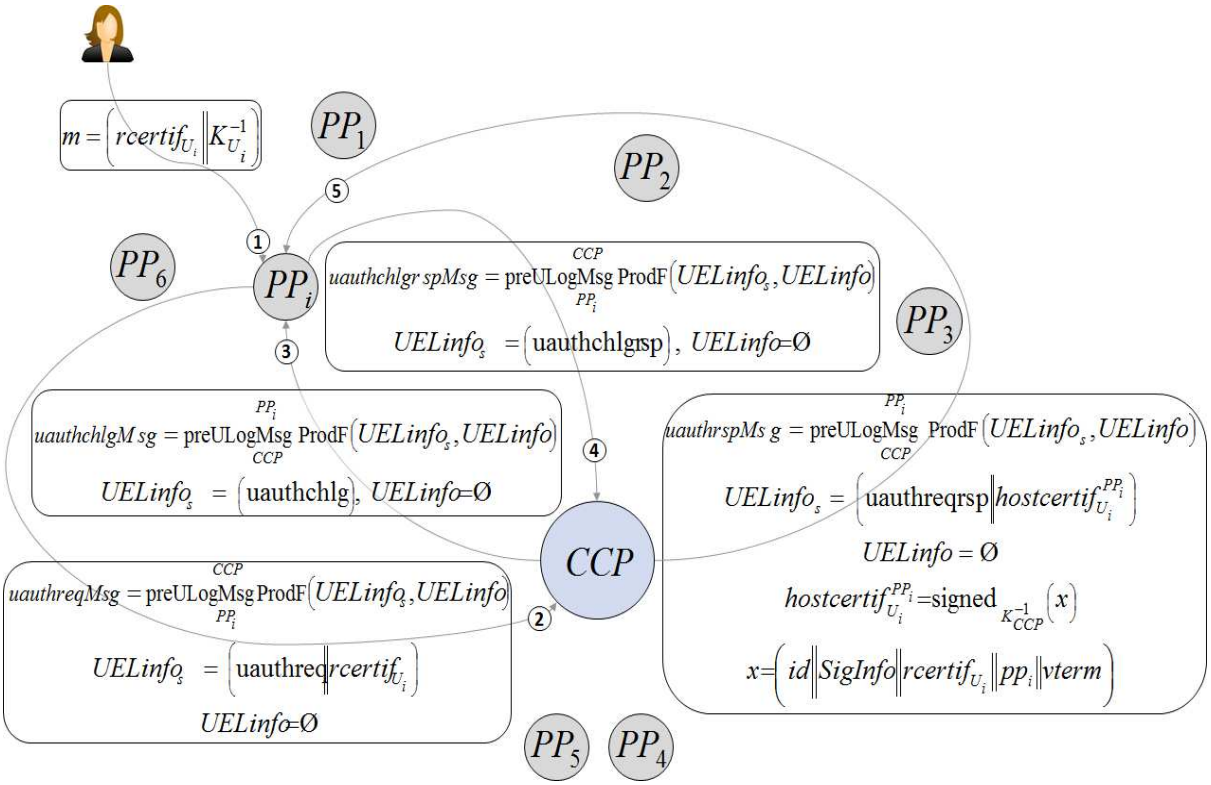


Figure 37 – User Authentication Procedure

After the authentication process is completed the $CCP$ performs the $OCP$ informing procedure explained in section 5.5.7.7.

### *5.5.6 User Action Monitoring*

User treatment of different MOs will vary. After having retrieved the MO from the system for local consumption, the consuming user, $U_i$, may for instance, watch it numerous times, only once or not at all. These different consumption behaviours indicate different preferences on the part of users.

Information pertaining to such user behaviours may, for instance, be employed for directed marketing or targeted advertising. Thus, in order to provide further support to the lateral extraction of gains, P2PTube also provides functionalities for the gathering of information on user behaviour.

Operations regarding the gathering and submission of this type of data, are performed at the peripheral peers (as these are the ones hosting the users), but are requested, (to the latter), by the $CCP$. The typical procedure is the following.

Whenever $CCP$ needs such information about user $U_i$, hosted at $PP_i$, it sends, to $PP_i$, a signed Event Report Request (ERR), (as the sensitive part of the UEL message), soliciting the monitoring of specific events on the part of $U_i$. $PP_i$ validates the ERR. If all is correct, $PP_i$ sends a signed acknowledgment to $CCP$ and proceeds to perform the requested monitoring.

If and whenever the targeted user action occurs, $PP_i$ prepares the corresponding Event Report (ER), signs it and sends it to $CCP$.
Further information on the nature and structure of the ERRs and ERs is available in section 7.4.9.

## *5.5.7 Management*

### *5.5.7.1 Introduction*

The proper operation of the UEL demands that some management procedures are undertaken periodically, or in response to some event, to assure the overall trustworthiness of the user collective.

These operations are centrally performed by the $CCP$, (based on notifications received form the periphery or outer core), which does all the necessary decision making, and translate into the issuing of $CCP$ instructions to $OCPs$ or $PPs$.

Said notifications may be generally described as $notif^{uel}_{P_i} = \text{signed}_{K^{-1}_{P_i}}\left(SigInfo \| notiftype \| notifcontent\right)$, where $SigInfo$ carries the necessary information to enable the validation of the notification, $notiftype$ designates the type of notification, and $notifcontent$ is the actual content of the notification.
The mentioned instructions may be generally described as $instruct^{uel}_{CCP}{}^{P_i} = \text{signed}_{K^{-1}_{CCP}}\left(SigInfo \| instructtype \| instructcontent\right)$, where $SigInfo$ carries the necessary information to enable the validation of the instruction, $instructtype$ designates the type of instruction, and $instructcontent$ is the actual content of the instruction.

The above introduced notifications and instructions are similar to those of the PLL (introduced in section 5.4.8.1). They have similar structure and are also signed by their emitting peers (hence, by the same keys as their PLL counterparts). However their logical role takes place at the UEL, and thus they are processed and exchanged at that level.

The main UEL management operations can basically be summed up as the pro-active diffusion of MOs throughout the peer tissue, the neutralization of infringing behaviour on the part of peers and users, and the updating of $OCP$ data structures.

### *5.5.7.2 MO Diffusion*

When MOs are first injected into the system, they are stored only at the $CCP$ and become available for redistribution. To optimize the distribution process, once an MO is successfully injected, the $CCP$ immediately seeds it throughout the peer tissue. This process consists of sending it (in a fragmented manner), to every $OCP$ and also to several $PPs$, all of which will then take part in the distribution of $MO_i^{CCP}$.

Furthermore, at periodical intervals, the $CCP$ evaluates the state of diffusion of all MOs, throughout the system's periphery. If it deems necessary, it delivers the relevant MOs to additional peripheral peers, which will then also start to redistribute them.

In order to "seed" an MO (e.g. $MO_i^{CCP}$), the $CCP$ must first prepare its fragmented version. It thus breaks the MO file into a number of fragments, of predetermined size, which it concatenates with their identifiers and then signs (the set) with its private key. $CCP$ then packages each generated fragment ($Frag_i^{MO_i^{CCP}}$) of $MO_i^{CCP}$, by producing its corresponding $Frag_i^{CCP\ MO_i^{CCP}} = \left( frag_i^{MO_i^{CCP}} \left\| Frag_i^{MO_i^{CCP}} \right. \right)$, where $frag_i^{MO_i^{CCP}}$ is the identifier of the fragment.

To seed that content at some specific peer $P_x$, $CCP$ then proceeds (in the secure manner defined in section 5.5.2.2 of the present chapter) to instruct $P_x$ to store and redistribute $MO_i^{CCP}$. Said instruction may be defined as $instrMOhold_{CCP}^{P_x\ uel} = \text{signed}_{K_{CCP}^{-1}} \left( SigInfo \| \text{instrMOhold} \| tstamp \| fraglist \right)$. It contains the adequate $SigInfo$ parameter, the instruction type definition, the time of the instruction's emission and $fraglist = \text{signed}_{K_{CCP}^{-1}} \left( id \left\| SigInfo \right\| mo_i^{CCP} \left\| fragdef_0^{MO_i^{CCP}} \right\| ..... \left\| fragdef_n^{MO_i^{CCP}} \right. \right)$. The latter carries the identifier of $fraglist$, its appropriate $SigInfo$ parameter, the identifier of $MO_i^{CCP}$ ($mo_i^{CCP}$), and the definition of each of the fragments ($fragdef_j^{MO_i^{CCP}} = \left( frag_j^{MO_i^{CCP}} \left\| size \right\| seqnr \left\| \left( SigInfo \left\| signature_{K_{CCP}^{-1}} \left( Frag_j^{MO_i^{CCP}} \right) \right. \right) \right. \right)$), into which the MO is divided for redistribution (for further information see section 5.5.8.2.2). The $fragdef$ parameters carry the identifier, the size, the sequence number (within the context of its encompassing MO), and the signature (by $CCP$), of their corresponding MO fragments. It contains, as well, the corresponding (to the latter signature) $SigInfo$ parameter. The instruction, in scope, is carried in UEL messages where $UELinfo_s = \left( \text{instrMOhold} \left\| instrMOhold_{CCP}^{P_x\ uel} \right. \right)$, and $UELinfo = \left( Frag_0^{CCP\ MO_i^{CCP}} \left\| .... \right\| Frag_n^{CCP\ MO_i^{CCP}} \right)$, which are acknowledged, by the destination peers, by sending back an UEL message where $UELinfo_s = (\text{instrshack})$.

### *5.5.7.3 Infringing or Faulty Peer Behaviour Neutralization*

The same problems addressed in section 5.4.8.3, within the context of PLL operation, also occur at the UEL level. The system shall address the issues at the UEL in the manner already described in section 5.4.8.3. The mains differences are:

- the procedure takes place at the UEL level, employing UEL messages;

- the employed misbehaviour notifications, sent from peripheral or outer core peers to the $CCP$, have the same structure as their UEL counterparts, but are designated as $notifmissbehav_{P_i}^{CCP}$ . They too are signed by their issuing peers;
- the system supports the reporting of misbehaviour in the same situation as those described in section 5.4.8.3, but dealing with UEL level information. It also reports requests which are not accompanied by the appropriate user hosting certificates or information retrieval permits.

### *5.5.7.4 Infringing User Behaviour Neutralization*

May different users, with varying degrees of honesty, participate in the system. There is thus a non-negligible possibility of some users incurring in infringing behaviour, which the system must be prepared to deal with.

This means that the $CCP$ must somehow obtain information about undesirable behaviour on the part of users. For this to occur, it must be users themselves to notify the $CCP$, whenever they feel that some other user is behaving inappropriately. Users may thus report:

- injection of inappropriate content;
- injection of corrupted content;
- injection of low quality content;
- injection of copyrighted content;
- re-injection of their own content by unauthorized user.

In all the above cases, the user hosting peer, $P_i$, will prepare a user misbehaviour notification, (on behalf of some user), defined as $notifmissbehav_{U_i}^{CCP} = \text{signed}_{K_{U_i}^{-1}}\left( SigInfo \| \text{missbehav} \| \left( id \| t \| u_i \| u_k \| infractDef \right) \right)$, where $id$ is the notification's identifier, $t$ is the notification's issuing time, $u_i$ is the identifier of the issuing (and thus, the signing), user (the user hosted at the peer), and $u_k$ is the identifier of the user which is targeted by the complaint. $infractDef$ is the definition of the alleged infraction, it may take the following values: $\left(\text{innaproMO} \| mo_i^{CCP}\right)$, $\left(\text{corruptMO} \| mo_i^{CCP}\right)$, $\left(\text{lowqualMO} \| mo_i^{CCP}\right)$, $\left(\text{copyrightMO} \| mo_i^{CCP}\right)$, and $\left(\text{appropriatedMO} \| mo_i^{CCP}\right)$. $notifmissbehav_{U_i}^{CCP}$ is signed by $U_i$, to enable the validation of its integrity and origin authenticity and assure un-repudiability of the report.

Once the notification is prepared, $P_i$ places it in the sensitive part of the UEL message ($UELinfo_s = \left( \text{notifUbehav} \| notifmissbehav_{U_i}^{CCP} \right)$) and proceeds to send it to $CCP$ (in the secure manner defined in section 5.5.2.3). $CCP$ then acknowledges it with a response UEL message where $UELinfo_s = \left(\text{notifUbehavack}\right)$.

The $CCP$ (with the assistance of human agents), periodically analyses all pending reports and decides what to do regarding the "criticized" users. It may do nothing, it may suspend or

it may expel the user. In the two latter cases the $CCP$ proceeds in accordance with what is described in section 5.5.7.5 of the present chapter.

On the other hand, the case of previously shunned users is also periodically revisited. In such moments $CCP$ may decide to let the user remain shunned, or re-enable its participation in the system. In the latter case it proceeds in accordance with what is described in section 5.5.7.6.

### 5.5.7.5 User Shunning

The system's users may, for whatever reason, engage in inappropriate or sabotaging behaviour. In such cases, once the $CCP$ detects such patterns (by way of notifications from users), it proceeds to shun the infringing user from the system either in a temporary or permanent base, and to place him in a quarantine or expulsion list, respectively.

To shun a specific user (e.g. $U_x$), the $CCP$ proceeds (in the secure manner defined in section 5.5.2.2 of the present chapter) to instruct all $OCPs$ to place $U_x$ in the user quarantine or expulsion list if $U_x$ was suspended or permanently banned, respectively. Said instruction may be defined as $instrushunning_{CCP}^{uel\ OCP_i} = \text{signed}_{K_{CCP}^{-1}}\left(SigInfo \| instrushunning \| (Q \text{ or } X \| u_x)\right)$, where the mentioned $OCP_i$ (in the equation), is the instructed $OCP$. They are carried in UEL messages where $UELinfo_s = \left(\text{instrsh} \left\| instrushunning_{CCP}^{uel\ OCP_i} \right.\right)$, which are acknowledged, by the destination peers, by sending back an UEL message where $UELinfo_s = (instrshack)$.

If it is possible to contact the shunned user (via its hosting peer), he will be sent, as well, one of the $instrushunning_{CCP}^{uel\ OCP_i}$ instructions, as a manner of informing him that he has been shunned or expelled.

From that point on, no requests from $U_x$ will be attended, until further notice. Any peripheral peer that attempts to login $U_x$ will fail.

### 5.5.7.6 User Readmission

To readmit a specific user (e.g. $U_x$), the $CCP$ proceeds (in the secure manner defined in section 5.4.2.2 of the present chapter), to instruct all $OCPs$ to remove $U_x$ from the user quarantine list. Said instruction may be defined as $instrureadm_{CCP}^{uel\ OCP_i} = \text{signed}_{K_{CCP}^{-1}}\left(SigInfo \| \text{instrureadm} \| (Q \text{ or } X \| u_x)\right)$, where the mentioned $OCP_i$ (in the expression), is the instructed $OCP$. They are carried in UEL messages where $UELinfo_s = \left(\text{instrsh} \left\| instrureadm_{CCP}^{uel\ OCP_i} \right.\right)$, which are acknowledged, by the destination peers,

by sending back an UEL message where $UELinfo_s = (\text{instrshack})$. From that point on, requests from $U_x$ will again be attended.

#### 5.5.7.7 OCP Updating

Whenever some change or addition needs to be made to the system's data structure (at the UEL level), it is the $CCP$ that coordinates that activity. The $CCP$ performs such changes on the version of the data structure which it locally stores, but it then becomes necessary to update all the other versions of the data structure, stored at the $OCPs$. Such an updating/initialization of an $OCP$'s data structure is also necessary when an $OCP$ first registers and connects to the system.

OCP updating, at the UEL, is in all similar to the same procedure at the PLL, which is described in section 5.4.8.6. The main differences are:

- the updated part of the data structure pertains to UEL aspects of the system and not to PLL ones;
- the procedure takes place at the UEL level, employing UEL messages;
- the employed updating instructions, sent from the $CCP$ to the relevant $OCPs$, have the same structure as their PLL counterparts, but are designated as $instrupdatedb_{CCP}^{uel\ \ OCP_i}$. They too are signed by $CCP$.

### 5.5.8 User Request Attending

#### 5.5.8.1 Introduction

User request attending operations may be divided into the following main categories, in terms of their overall complexity:

- Simple Operations – these operations perform the input or retrieval of some individual piece of information (e.g. the insertion of an MO, the retrieval of an MO, the insertion of a specific monetary amount into a user's account, etc.). They may further be subdivided, in what pertains to their implications on the system's global data structure, into:
  - Writing Operations – these operations imply the performing of changes to the system's global data structure (e.g. monetary transactions, insertion and removal of MOs, etc.). They are of the client/server type;
  - Reading Operations – these operations imply only the reading of the system's global data structure (such as the semantics based searching for MOs and the retrieval/distribution of MOs, among others). These operations will typically unfold in a hybrid P2P way;
- Composed Operations – these operations are composed of simple and/or other composed operations. They may thus have client/server and hybrid P2P parts.

#### 5.5.8.2 Simple User Attending Operations

##### 5.5.8.2.1 Simple Writing User Attending Operations

The handling of all user requests that imply a simple type of operation, and some change or addition to the system's global data structure, will require the participation, (besides other peers), of the $CCP$ as it is the gatekeeper to that structure. The handling of such user requests will thus be performed through client/server operations at the UEL.

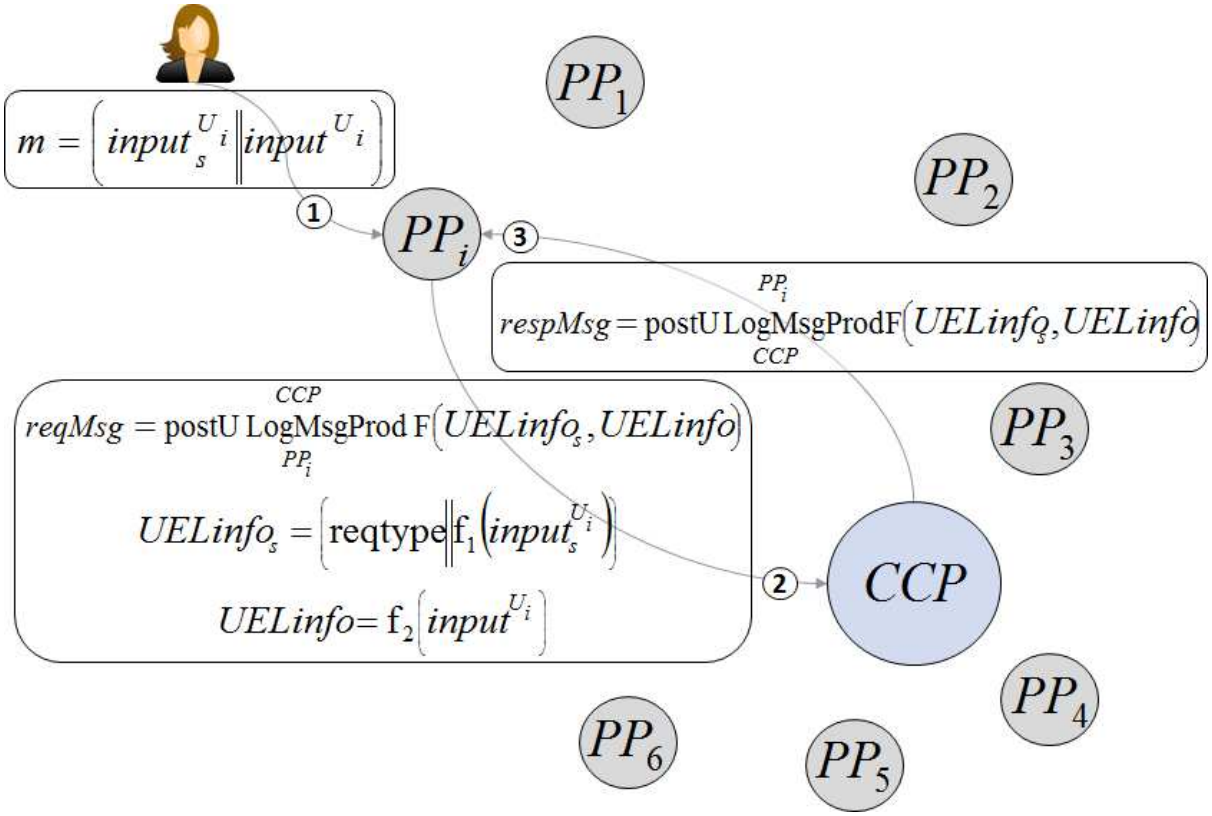


Figure 38 – Client/Server User Attending Procedure

Requests included in this group are those involving the following activities:

- user abandonment of the system;
- injection, removal, updating and versioning of MOs;
- maintenance of user accounts and performing of currency transactions;

User registration and login are also handled through simple, client server operations, but in the ways presented in sections 5.5.3 and 5.5.5 respectively.

The typical simple client/server UEL operation is depicted in Figure 38, (taking into consideration the notation defined in section 5.1).

$U_i$ delivers the sensitive and non-sensitive input parameters that make up its request, to his hosting peer, $PP_i$ (step 1). $PP_i$ builds $UELinfo_s$ by concatenating the request type designation with the sensitive input parameter (or some function, $f_1$, of it). $PP_i$ builds, also, $UELinfo$ as some function, $f_2$, of the non-sensitive user input information. $f_1$ and $f_2$ represent possible changes or additions that the peer may need to operate on the received parameters before sending them.

Given what is exposed in section 5.5.2.3, the UEL message will contain the user hosting certificate and will be signed with the user's private key. That signature, the presence of this certificate, and the indication, (that said certificate carries), of the hosting peer, enables the receiving end to verify, (employing also the PLL delivered information about the message's validated sending peer, mentioned in section 5.4.9), if the message sending peer is in fact the, $CCP$ recognized, host of the user in scope, and thus, is authorized to issue the request at stake.

$PP_i$, then, securely sends the produced UEL message to $CCP$ (step 2), though the services of the PLL. The sensitive part of the UEL message is carried as the sensitive part of the PLL message, while the non-sensitive part of the earlier message is carried as the non-sensitive part of the latter.

$CCP$ produces the adequate response and securely sends it to $PP_i$ (step 3), which displays the relevant output to $U_i$.

After the user request has been handled, the $CCP$ takes care of propagating the changes/additions, which were made to the system's data structure, to the concerned $OCPs$ (in the manner described in section 5.5.7.7.).

The following sub sections present some examples of client/server user attending operations.

#### *5.5.8.2.1.1 MO Insertion Operation*

$PP_i$ builds $UELinfo_s = (\text{moinsertionreq})$ and $UELinfo$ . The latter object is a function of the non-sensitive input parameter, $input^{U_i}$ . This parameter consist of all the necessary data objects to build an MO. $UELinfo$ is, thus, the assembled MO, (identified as $mo_i$), which $PP_i$ also signs with the user's private key (exceptionally, in order to enable a later independent treatment of $MO_i$ while preserving the bond that connects it to the author user).

The UEL message is then sent to $CCP$ . $CCP$ formally validates the MO and it may also perform some automated or user assisted content analysis to determine if the inserted MO does not violate the system's terms of use (respects copyright, etc.). If all is ok, the $CCP$ accepts $MO_i$ , and proceeds to its installation in the system. It thus signs $MO_i$ , with its own private key, to signal the system's acceptance of that object's insertion into it, thus producing $MO_i$'s delivery-ready version, $MO_i^{CCP}$ . *CCP* then produces the fragmented version of $MO_i^{CCP}$ (see section 5.5.7.2).
After that, $CCP$ notifies $PP_i$, of the acceptance of $MO_i$ , with an UEL response message which carries, in its non-sensitive part, the fragmented version of $MO_i^{CCP}$ .
Finally $CCP$ performs the initial seeding of $MO_i^{CCP}$ (see section 5.5.7.2), through the system's tissue, by sending it (in a fragmented manner), for storage and redistribution, to some selected peripheral peer(s).

Furthermore, $CCP$ would also take care of propagating the changes/additions, which were made to the system's data structure, to the $OCPs$ . Thus, it would take care of divulging:

- the $MO_i^{CCP}$ and all the relevant information pertaining to it;
- the information that $U_i$ is the owner of $MO_i^{CCP}$ ;
- the identification of all the peripheral peers which are storing $MO_i^{CCP}$ , and ready to redistribute it;
- the information regarding any relationships between $MO_i^{CCP}$ and other MOs or other operational objects.

#### 5.5.8.2.1.2 Monetary Resource Insertion Operation

In a simplistic manner this operation may be described as follows. $U_i$ requests from the system a specific code to enable him to transfer some amount of many to his account in the system. $PP_i$, thus, builds $UELinfo_s = (\text{loadaccntreq})$ , and, proceeding in accordance with what was explained in section 5.5.2.3, sends a UEL request message to the *CCP*. The latter peer validates the received message and if all is ok sends back an UEL message where $UELinfo_s = (\text{ackloadaccntreq} \| \text{specfcode})$ , and specfcode is the specific reference for the user to employ, in the inter-banking transfer, to load his system account.

$U_i$ then proceeds to the out-of-band money transfer, by whatever means he prefers (typically employing the system's publicly know bank account number and the given reference). Once the system (the *CCP*) detects that transfer, it proceeds to load the user's system account with the received amount.

From that point on the loaded monetary resources are available for the user to employ, within the system as he wishes.

### 5.5.8.2.2 Simple Reading User Attending Operations

The handling of user requests that imply simple operations and no changes or additions to the system's data structure, may not require the participation of the $CCP$ , as much of the information stored in it may also be accessed through the services of the $OCPs$ or of $PPs$ . The handling of such user requests will thus be performed through a mixed P2P and client/server operation at the UEL.

Requests included in this group are those involving the following activities:

- the semantics based searching for MOs;
- the retrieval/distribution of MOs;
- the searching and retrieval/distribution of information objects pertaining to users or other aspects of the system.

To explain the inner workings of this type of system operations, the next sub-section presents he MO retrieval operation.

#### 5.5.8.2.2.1 MO Retrieval Operation

The typical system procedure to retrieve a specific MO object, $MO_i^{CCP}$ , is depicted in Figure 39, (taking into consideration the notation defined in section 5.1).
$U_i$ informs her hosting peer, $PP_i$, that she wishes to consume $MO_i^{CCP}$ (step 1). $PP_i$ then sends an UEL request for $MO_i^{CCP}$ , to $OCP_i$ (step 2). That request message contains (as its sensitive part), the designation of the type of request ($requelinfo$) and the identifier of the desired MO.

$OCP_i$ prepares an UELInfo retrieval permit ($retrpermit_{uelInfo}^{PP_i}$), enabling $PP_i$ to perform the retrieval of $MO_i^{CCP}$ from a number of other peripheral peers. The permit contains its identifier, the designation of the permit's type (uelInfoRetPerm), the $SigInfo$ parameter which contains the necessary data to enable the validation of the permit's signature, the identification of the peer to which it has been emitted ($pp_i$), the specification of the permit's

validity term date, the definition of the type of information that may be retrieved ($infoDef = \left(mo_i^{CCP}\right)$), and the identifier of the $uelildo_s$ object (see section 7.4.5).

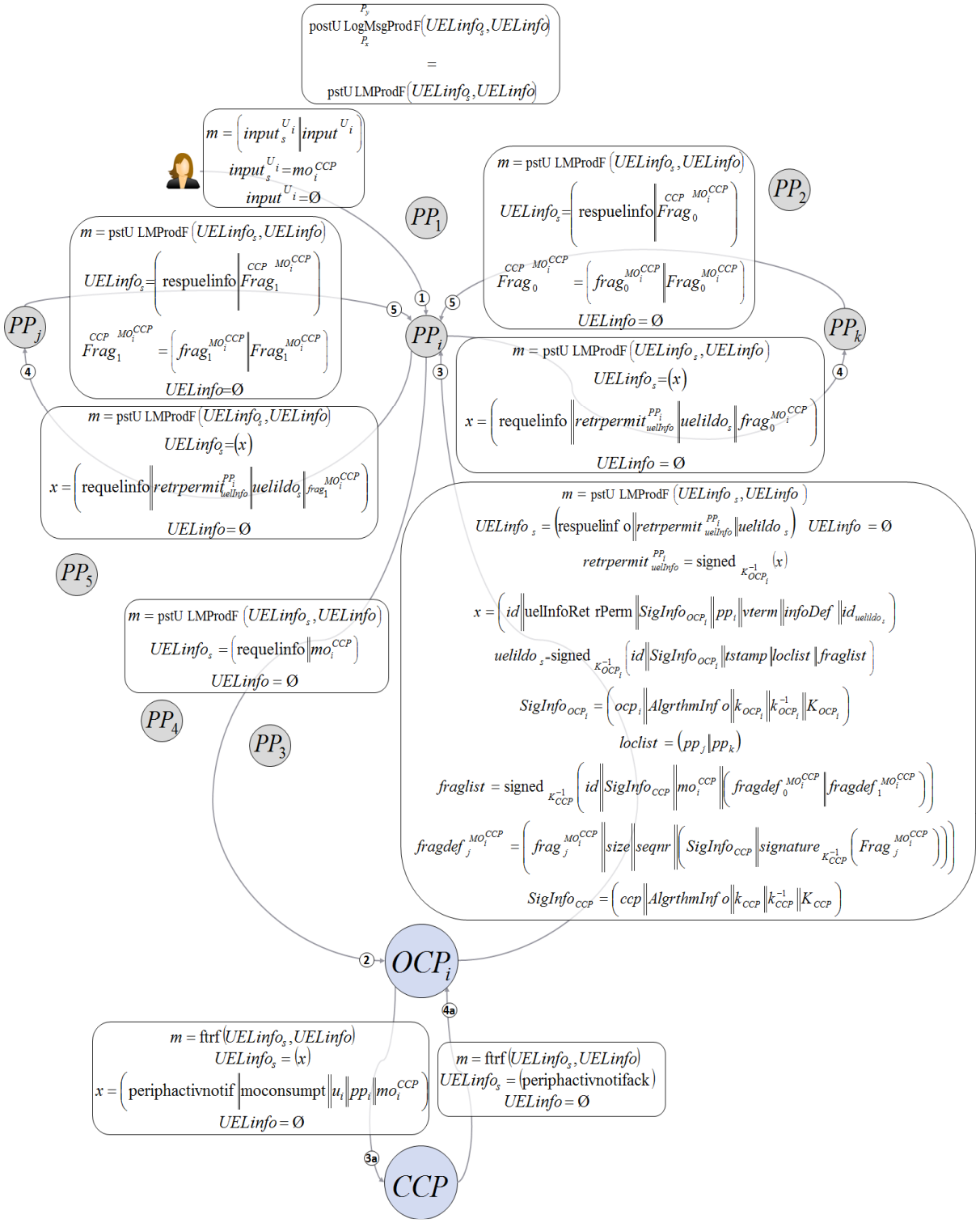


Figure 39 – Hybrid Operation for MO retrieval

The UEL Information Location Describing Object ($uelildo_s$), carries (in this specific case of the retrieval of an MO), its identifier, the $SigInfo$ parameter which contains the necessary

data to enable the validation of $uelildo_s$'s signature, the *fraglist* object, a list of the peers ($loclist$), from which $PP_i$ may retrieve $MO_i^{CCP}$ (or fragments of it), and a timestamp indicating the time at which $uelildo_s$ was compiled (so that peers using that information may know how old it is).

*fraglist* contains, its identifier, its adequate $SigInfo$ parameter, the MO's id ($mo_i^{CCP}$) and the $fragdef_j^{MO_i^{CCP}}$ objects that pertain to the fragments into which the MO was divided to enable its fragmented redistribution. Each $fragdef_j^{MO_i^{CCP}}$ carries the identification, size, sequence number and signature, (by the $CCP$), of a specific fragment of $MO_i^{CCP}$, and the corresponding (to said signature) $SigInfo$ parameter. $fraglist$ is signed by the $CCP$ to assure its integrity and authenticity, as it was that peer ($CCP$) that produced that object at MO insertion time.

$retrpermit_{uelInfo}^{PP_i}$ and $uelildo_s$ are signed by $OCP_i$ (their issuing $OCP$), to enable the validation of their authenticity, by any peer. The independent and redundant signing of $retrpermit_{uelInfo}^{PP_i}$ and $uelildo_s$, (in regards to the sensitive part of the PLL message that wraps the UEL message that contains them), enables the later independent P2P distribution of $uelildo_s$ amongst the peripheral peers and the presentation of $retrpermit_{uelInfo}^{PP_i}$, by, $PP_i$, to other peers to prove the authorization of its requests.

In the next step of this operation, $OCP_i$ sends $retrpermit_{MO_i^{CCP}}^{PP_i}$ to $PP_i$ (step 3). After that, $PP_i$ proceeds (in a P2P fashion), to simultaneously retrieve $MO_i^{CCP}$'s composing fragments, from its delivering peers (e.g. $PP_j$ and $PP_k$). It sends messages to $PP_k$ and $PP_j$ (steps 4), requesting fragments $Frag_0^{MO_i^{CCP}}$ and $Frag_1^{MO_i^{CCP}}$ respectively. In such messages it includes, (besides the designation of the type of request and the identification of the desired fragment), the permit and $uelildo_s$ objects, received from $OCP_i$, so that the $PP_j$ and $PP_k$ can verify that $PP_i$ is indeed authorized to retrieve the requested MO fragment from them (the permit carries the identifier of the $uelildo_s$ object, and so the connection between the two is established and may be verified).
Finally $PP_k$ and $PP_j$ deliver said fragments (steps 5), to $PP_i$ which validates their integrity (employing the information present in the $uelildo_s$ object), reconstitutes $MO_i^{CCP}$, validates its overall integrity (as it carries its signature by the *CCP*) and renders it for user consumption.

After it has answered $PP_i$'s request, $OCP_i$ reports to $CCP$, that $U_i$ is consuming $MO_i^{CCP}$ at $PP_i$ (step 3a). $CCP$ then takes care of registering such information in the system's data structure and of propagating those changes, to the data structure, to the $OCPs$.

This overall procedure thus enables an efficient P2P diffusion, of MOs, between peripheral peers. It may also be applied to other UEL information objects as user registration certificates ($rcertif_{U_i}$), user hosting certificates, ($hostcertif_{U_i}^{PP_i}$), content search response objects (see *sqro* in section 7.4.4), or UEL Information Location Describing Objects ($uelildo_s$ s).

### *5.5.8.3 Composed User Attending Operations*

To explain the inner workings of composed user attending operations we present bellow an example of such an operation.

#### *5.5.8.3.1 User Attention Sale Operation*

As explained in section 2 of chapter IV, two main types of user attention sale procedures are to be supported by the P2PTube architecture. These are: mandatory unrewarded advertisement viewing; and voluntary rewarded advertisement viewing.

The latter procedure is supported as a set of UEL operations, which may be described in the following manner:

- At an initial time, a specific advertising user, $U_{add}$, inserts into the system a number of advertising MOs and their corresponding Inquiry IOs and Inquiry Response IOs (see section 7.4.10). Amongst these is $MO_{add}^{CCP}$, one of its corresponding inquiry IO, $inquiryIO_i^{CCP}$ and its respective Inquiry Response IO $inquiryRespIO_i^{CCP}$. These MO and IO insertion operations are supported in the client/server manner indicated in section 5.5.8.2.1;
- Some time later, a regular user, $U_i$, decides to "sell" some attention. For that she selects a list of available advertisement MOs from the system by performing a query. This operation is supported in a hybrid manner as explained in section 5.5.8.2.2;
- $U_i$ then selects her preferred advertisement MO (e.g. $MO_{add}^{CCP}$), requesting its retrieval – this retrieval operation unfolds in the hybrid manner explained in section 5.5.8.2.2;
- $U_i$ then retrieves $inquiryIO_i^{CCP}$, (in the hybrid manner explained in section 5.5.8.2.2), and answers its questions, pertaining to the contents of $MO_{add}^{CCP}$. Said answers are then sent back to the *CCP*. The latter checks them against the information present in $inquiryRespIO_i^{CCP}$. If all is ok, a specific monetary amount is extracted from $U_{add}$'s account and a portion of it is deposited in $U_i$'s account. Another portion is deposited in the system's operating entity's account.

The earlier procedure may be supported by the following set of UEL operations:

- The initial step is the same as in the previous bullet list;
- Some time later, a regular user, $U_i$, selects some specific MO (e.g. $MO_i^{CCP}$) for consumption, which is retrieved in the hybrid manner explained in section 5.5.8.2.2;
- In its rights expressing metadata $MO_i^{CCP}$ indicates that it should only be given access to, after the user consumes a specific advertisement MO (e.g. $MO_{add}^{CCP}$). Thus, the user hosting peer ($PP_i$) retrieves $MO_{add}^{CCP}$, from the system, in the hybrid manner explained in section 5.5.8.2.2;

- $PP_i$ then renders $MO_{add}^{CCP}$ and only after that does it render $MO_i^{CCP}$;
- After $MO_{add}^{CCP}$ has been rendered, $PP_i$ informs $CCP$ (in the client server manner explained in section 5.5.8.2.1), with a signed (with the user's private key) notification, of the advertisement consumption activity that has occurred;
- $CCP$ then proceeds to extract a specific monetary amount from $U_{add}$'s account and deposit it in the system's operating entity's account.

### *5.5.9 DRM Enforcement*

The enforcing of DRM-like aspects is not handled in any isolated component or operations in the system. Instead said enforcement is logically included in numerous operations that take place within the system.

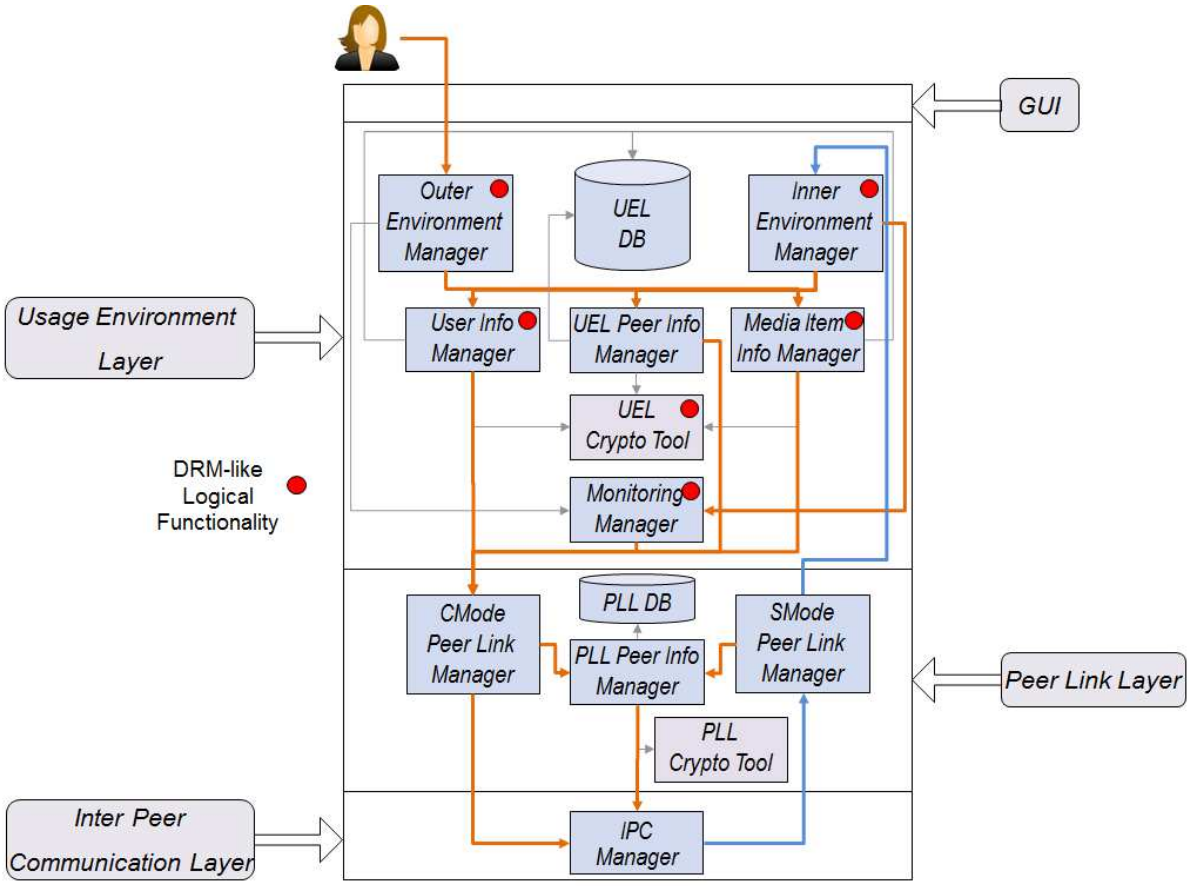


Figure 40 – Logical DRM Functionalities in P2PTube Peer Structure

All UEL (and the related PLL) security related measures are devoted to guarantying communicational confidentiality, data integrity, data origin authenticity and non-repudiability and user and peer identity. They thus contribute to a reliable and trustworthy identification of users, peers and media content, to the preservation of MO integrity and to the protection of the association between users and the media content that they own.
Said measures, thus, protect the right of all users to privacy and to a secure access to their account and user data, and protect the rights of content owners as they guard against content corruption and wrongful ownership declaration (by way of user reports).

The mechanisms supported, by the system, for the reporting, by system users, of user misbehaviour (described in section 5.5.7.4), basically consist of a distributed collaborative system for the detection of content whose authorship is being wrongfully claimed, that is, for

the detection of copyright infringement. Given the overall collaborative nature, intended for the system, and its interdependency-based business models, it is expectable that users will proactively aid in the detection of copyright infringement, and thus, this mechanism should be successful.

Furthermore, the user action monitoring capabilities of the system (at the UEL), also enable the gathering of information on user behaviour with implication on the protection of content owner rights. The collected information may also be used for the purpose of making monetary gains, by the system's operating entity, thus supporting its right to be rewarded for its operation.

Revisiting Figure 25 (from section 3.3 of the present chapter), which describes the structure of a peer, one may thus visualize the DRM capacities of the system, as being logically distributed over the set of components signalled with a red circle in Figure 40.

In accordance with the type of business models that are to be supported, the systems, thus, employs "light" DRM capabilities.

# 6 Inter-System Cooperation

## 6.1 Overview

Systems, embodying this architecture, may cooperate with one another. Said architecture provides the necessary means to assure operational reliability and communicational security and confidentiality and informational authenticity. It achieves this through the employment of semi-centralized cryptographic means which are to be under the control of a trustworthy entity.

The provision of the above stated guaranties, to peers and users, in the context of inter-system interaction demands the existence of a solid context of trust between the interacting systems. Users and peers of system $Sys_A$ can only interact with users and peers of system $Sys_B$, if $Sys_A$ trusts $Sys_B$, that is, if the operating entity of the earlier system trusts the operating entity of the latter one. If this trust relationship, which can only be built through inter-human interaction in the real world, does not exist, the two systems cannot interact.

Furthermore, for the correct operation of these systems, the trust relationships between them must be reciprocal. For instance, for a peer, $P_A$, from system $Sys_A$, to be able to cooperate with some peer, $P_B$, in system $Sys_B$, $P_A$ must know and trust the authentication information of $P_B$ and vice-versa. For said peers to be able to trust that information, about each other, $Sys_A$ must trust $Sys_B$ and vice-versa.

The cooperation, between any two such systems, implies, thus, the existence of a mutual trust between their operating entities.
The cooperation networks, collectively established by these systems, are therefore constituted as a form of web of trust [174]. That network is composed by a set of bilateral trust relationships between individual systems, where each system, in the relationship, will trust (and therefore cooperate with), the other system.

In a typical web of trust approach if entity *A* trusts entity *B*, and *B* trusts entity *C*, then, it follows that *A* trusts *C*, that is, trust is a transitive relationship. However, given the demanding security requirements for a safe operation of the P2PTube architecture, in the context of the trust networks established between these systems, that relationship must not

be transitive. Thus, a P2PTube system trusts and cooperates only with the homologue systems that its operating entity directly instructs it to trust, and no others.
The set of all (reciprocally) trusted systems of some specific system is defined as the latter's trust sphere.

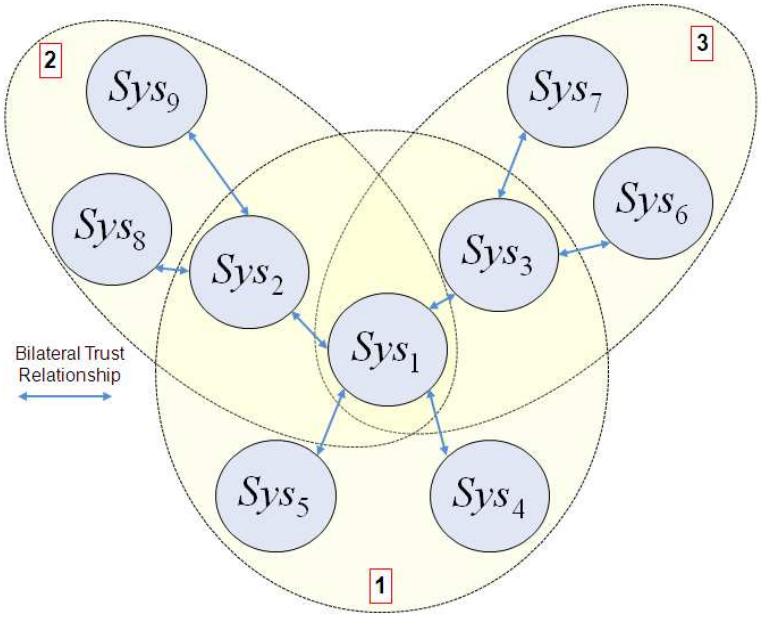


Figure 41 – Inter System Trust Spheres Example

Figure 41 presents an example of a possible trust context maintained by a set of 9 independent systems. In the exemplified context three different trust spheres are highlighted:

- The trust sphere of system 1 – includes all the systems with which it maintains a bidirectional trust relationship, that is, system 2, system 3, system 4 and system 5;
- The trust sphere of system 2 – includes systems 1, 8 and 9;
- The trust sphere of system 3 – includes systems 1, 6 and 7;

As it can be seen, no transitivity exists, in inter-system trust relationships, as, for instance, while system 6 is part of system 3's trust sphere, and the latter is part of system 1's trust sphere, system 6 is not part of system 1's trust sphere.

The establishment of a mutual trust relationship between two P2PTube systems is involves the active participation of their operating entities. These entities must configure their systems' *CCPs* to trust each other. Each *CCP* will then diffuse information describing and validating the established trust relationship throughout the system's tissue.

## 6.2 Trust Relationship Establishment

The inter-system trust establishing process is performed at the UEL, and is exemplified in Figure 42. To initiate it, the operating entity of one of the systems takes the initiative to instruct its system's *CCP*, $CCP_A$, to trust the *CCP* of the target system, $CCP_B$. To do so, said entity supplies its peer the identification, public key and present IP address of $CCP_B$ (step 1).

$CCP_A$ then prepares and signs the trust relationship describing info object, $trustrel$. This contains the identification, public and private key identifiers and the public key of each of the *CCPs* of both systems. It carries also a timestamp indicating the validity termination date of

the described trust relationship. The signed version of *trustrel* (concatenated with the adequate *SigInfo*), is then sent to $CCP_B$ (step 2).

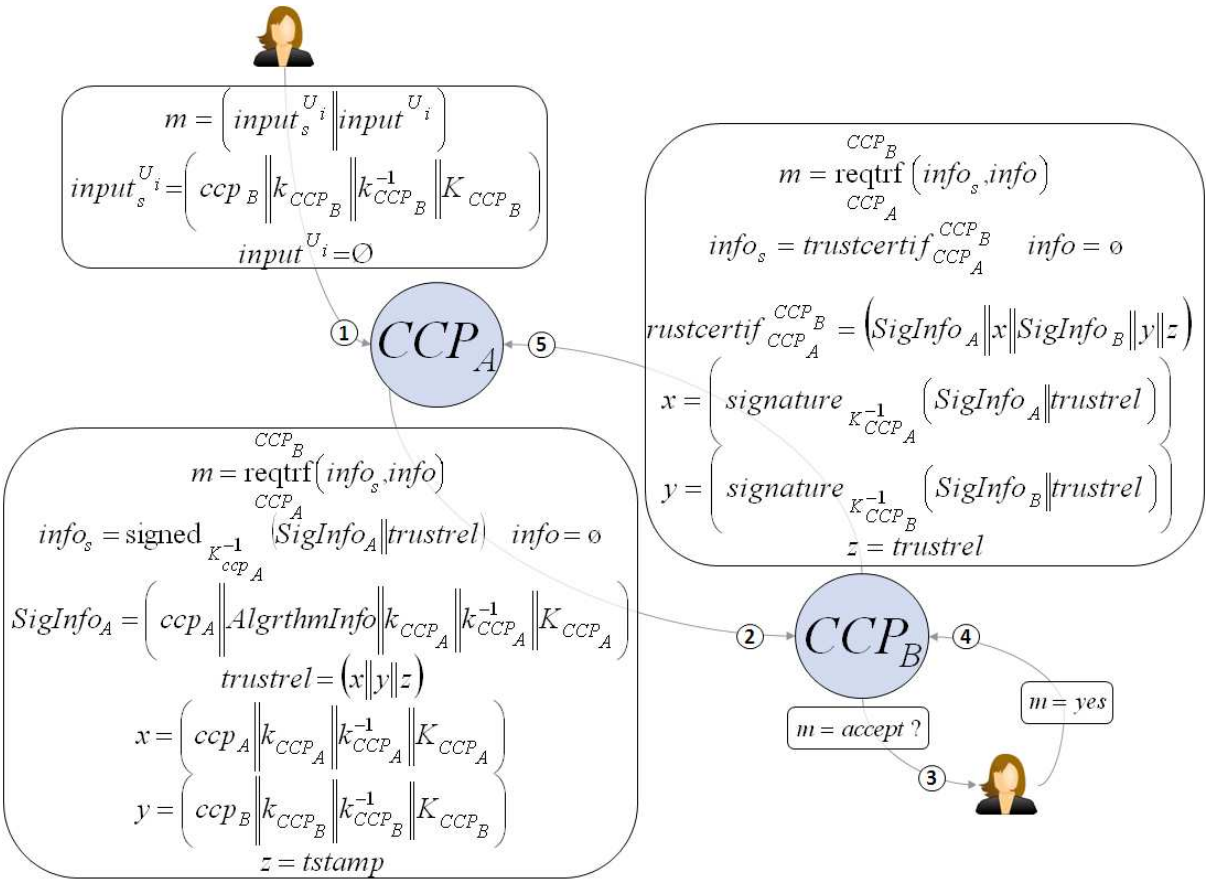


Figure 42 – Inter-System Trust Establishment

In order to accept or refuse the received request to establish a trust relationship, $CCP_B$ must obtain the input of its operating entity. For that reason it requests (step 3), from said entity, an authorization to establish a trust relationship between the two systems.
If system B's operating entity does accept what was requested (step 4), $CCP_B$ produces the trust certificate object, $trustcertif_{CCP_A}^{CCP_B}$. This object contains the signatures of the $(SigInfo \| trustrel)$ set, by $CCP_A$ and $CCP_B$, and *trustrel* itself. The two signatures prove the mutual acceptance of the trust relationship. $trustcertif_{CCP_A}^{CCP_B}$ is then sent to $CCP_A$, (step 5), and the trust relationship is established.

## 6.3 Trust Information Diffusion

Whenever a peer, $P_A$, from system $Sys_A$ needs to interact (either at the PLL or at the UEL), with a peer, $P_B$, from another system, $Sys_B$, it must first assess if the target peer's system is trusted or not. To do so, $P_A$ must obtain from its *CCP*, ($CCP_A$), a list of all the trust relationships in which its system participates with other systems ($trustlist_A = \left( sys_1 \| trustcertif_{CCP_A}^{CCP_1} \right)_{\| (.....) \|} \left( sys_n \| trustcertif_{CCP_A}^{CCP_n} \right)$). This object is a list of the trust

certificates of all the trust relationships in which the $Sys_A$ participates along with the identification of every trusted system.

In possession of this information, $P_A$ may determine if $P_B$ is part of a trusted system or not. This is possible because every peer's identifier carries, as a prefix, the identification if its encompassing system (as explained in section 5.2.3).

# 7 Data and Metadata Objects

## 7.1 Introduction

The regular operation of the system involves the production and exchange/distribution of a set of different information objects. To assure systemic reliability, these must be structured so as to guarantee their integrity, authenticity and temporal validity.
In P2PTube, MPEG-21 is employed for the definition of most such objects, given the information structuring capabilities of that standard. The most relevant of these objects are presented in the following sub sections of section 7.

## 7.2 IPCL Objects

### *7.2.1 IPCL Message*

IPCL messages (the messages exchanged between IPCL instances in different peers), consist of TAR archives which have the following content:

- IPCL Message Head File (*IPCLMHFile*) – contains the IPCL message's metadata;
- PLL Message File (*PLLMFile*) – consists of the PLL message which is being wrapped by the IPCL message.

The contents of the *IPCLMHFile* may be defined as $ipclMHFile = \left( srcPIP \| dstPIP \| pllMFileRef \right)$, where $srcPIP$ and $dstPIP$ are the IP addresses of the source and destination peers, respectively, and $pllMFileRef$ is the ref to the PLL message carried inside the IPCL message file. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 43.

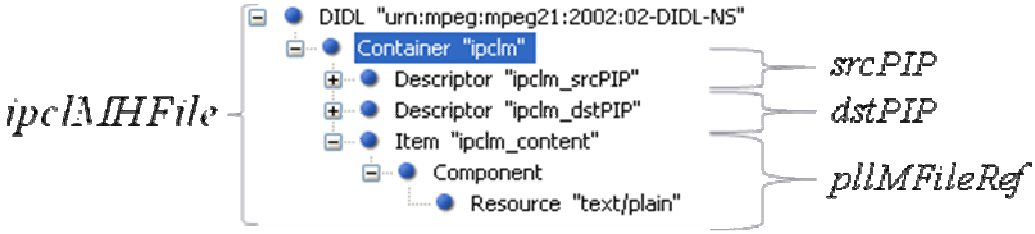


Figure 43 – IPCLMHFile Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the head file's contents. This latter element carries:
  - a *Descriptor* element containing the IP address of the sending peer;
  - a *Descriptor* element containing the IP address of the destination peer;
  - an *Item* element (identified as *ipclm_content*), which carries a reference to the PLL message file which is wrapped inside the IPCL message object.

An example of an *IPCLMHFile* may be found in section A.1.1 of Annex A.

## 7.3 PLL Objects

### *7.3.1 PLL Message*

In section 5.4.2 a PLL message (sent from $P_i$ to $P_k$) is logically defined as $\left(\left(\left(EncInfo \| \text{enc}_{K_{P_k}}\left(\text{signed}_{K_{P_i}^{-1}}\left(SigInfo \| serial \| Info_S \| \text{signature}_{K_{P_i}^{-1}}(Info)\right)\right)\right) \| (Info)\right)\right)$, where

$EncInfo = \left(\left(p_i \| AlgrthmInfo \| k_{P_i} \| k_{P_i}^{-1} \| K_{P_i}\right) \cup s_g^{P_i - P_k}\right)$, $SigInfo = \left(p_i \| AlgrthmInfo \| k_{P_i} \| k_{P_i}^{-1} \| K_{P_i}\right)$.

In more precise terms they consist of TAR archives with the following content:

- PLL Sensitive Info File (*PLLSIFile*) – contains the PLL message's sensitive information. It consists of a Tar archive containing:
  - PLL Sensitive Info Head File (*PLLSIHFile*) – contains the top, unprotected metadata of the *PLLSFile*;
  - PLL Protected Sensitive Info File (*PLLPSIFile*) – a TAR archive, (encoded with $K_{P_k}$), containing:
    - PLL Protected Sensitive Info Head File (*PLLPSIHFile*) – contains the signature of the content of the following file;
    - PLL Protected Sensitive Info Core File (*PLLPSICFile*) – contains the signed part of the *PLLPSIFile*;
- None or one PLL Non Sensitive Info File (*PLLNSIFile*) – carries non sensitive PLL information. It may wrap a UEL Non Sensitive Info File (*UELNSIFile*).

The *PLLSIHFile* (which basically contains the $EncInfo$ part), consists of an MPEG-21 DID with the structure depicted in Figure 44. An example of the contents of a *PLLSIHFile* may be found in section A.2.1.1 of Annex A.

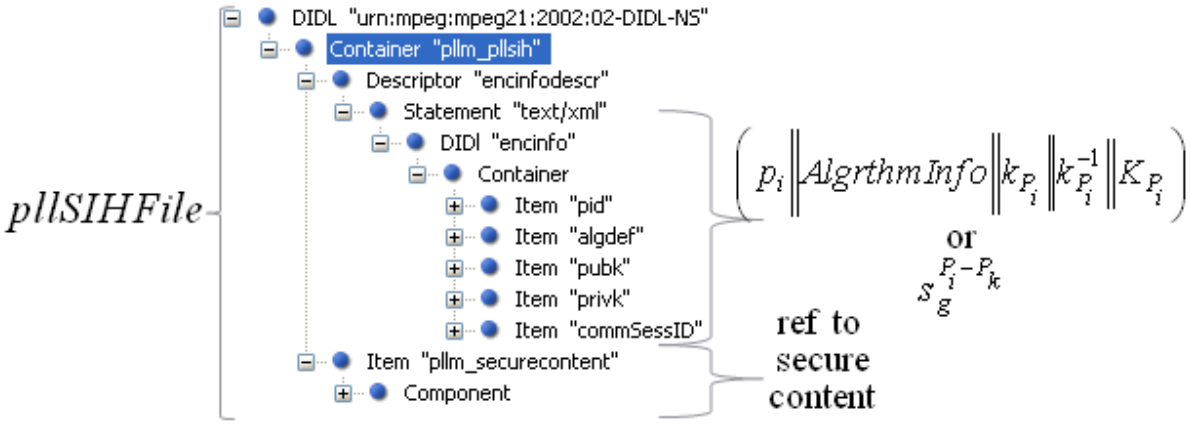


Figure 44 – PLLSIHFile Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the file's contents. This latter element carries:
  - one of the following two options:
    - a *Descriptor* element containing the *EncInfo* parameter. This element carries a DIDL which may contains inside:
      - either:

- an *Item* element which carries the identifier of the peer that performed the encoding;
- an *Item* element which carries the definition of the encoding algorithm;
- an *Item* element which carries the peer's public key and its identifier;
- an *Item* element which carries the identifier of the peer's private key;

- or:
  - an *Item* element which carries the identifier of the communication session of which this PLL message is a part;

- an *Item* element (identified as *pllm_securecontent*), which carries a reference to the file containing the ciphered contents of the PLL message.

The *PLLPSIFile* consists of $\text{enc}_{K_{P_k}}\left(\text{signed}_{K_{P_i}^{-1}}\left(SigInfo \| serial \| Info_S \| \text{signature}_{K_{P_i}^{-1}}(Info)\right)\right)$.

The *PLLPSIHFile* (which contains $\text{signature}_{K_{P_i}^{-1}}\left(SigInfo \| serial \| Info_S \| \text{signature}_{K_{P_i}^{-1}}(Info)\right)$), consists of an MPEG-21 DID with the structure depicted in Figure 45.

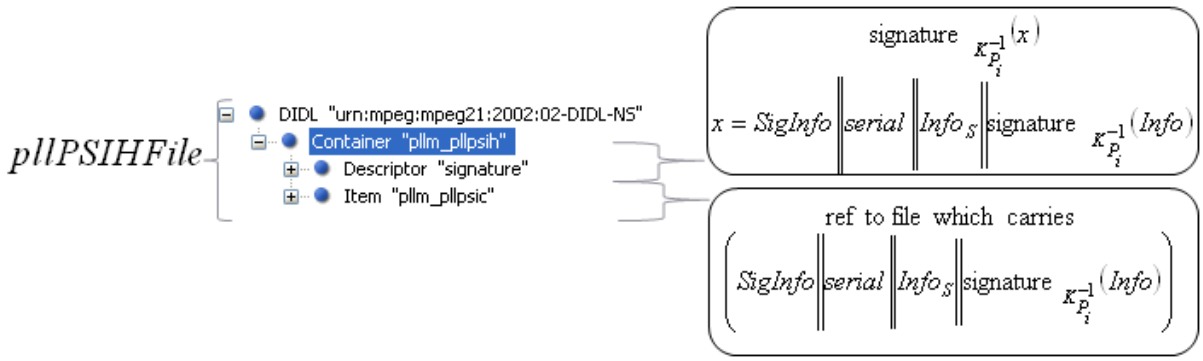


Figure 45 – PLLPSIHFile Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the head file's contents. This latter element carries:
  - a *Descriptor* element containing the signature of the file referenced in the item bellow;
  - an *Item* element (identified as *pllm_pllpsic*), which carries a reference to the PLL Protected Sensitive Info Core File (*PLLPSICFile*);

An example of the contents of a *PLLPSIHFile* may be found in section A.2.1.2 of Annex A.

The *PLLPSICFile* is a Tar archive containing the following files:

- The *PLLPSIC* Top File (*PLLPSICTopFile*) – carries $(SigInfo \| serial)$;
- The *PLLPSIC* Middle File (*PLLPSICMiddleFile*) – carries $Info_S$;

- The *PLLPSIC* Bottom File (*PLLPSICBottomFile*) – carries $\left(\text{signature}_{K_{P_i}^{-1}}(Info)\right)$.

The *PLLPSICTopFile* (which contains $\left(SigInfo \,\|serial\,\right)$), consists of an MPEG-21 DID with the structure depicted in Figure 46.

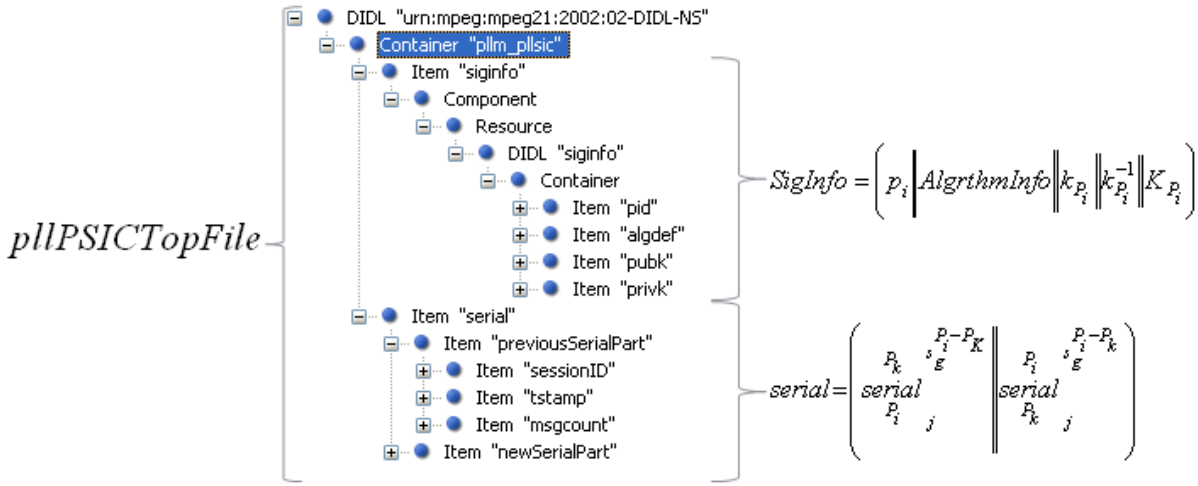


Figure 46 – PLLPSICTopFile Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the file's contents. This latter element carries:
  - an *Item* element (identified as *sigInfo*), which carries the serial of *SigInfo* parameter;
  - an *Item* element (identified as *serial*), which carries the serial of the PLL message;

An example of the contents of a *PLLPSICTopFile* may be found in section A.2.1.3 of Annex A.

The *PLLPSICBottomFile* (which contains $\left(\text{signature}_{K_{P_i}^{-1}}(Info)\right)$), consists of an MPEG-21 DID with a structure consisting of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the file's contents. This latter element carries:
  - an *Item* element (identified as *nonsensitiveinfosig*), which carries the signature, (by the message sending peer) of the file carrying the $Info$ part of the PLL message;

The *PLLPSICMiddleFile* is a Tar archive that carries a concatenation of the following files:

- *PLLPSICMiddle* Metadada File (*PLLPSICMiddleMDFile*) – carries the PLL level metadata of $Info_S$;
- None or one UEL Sensitive Info File (*UELSIFile*) – this file consists of the sensitive part of the UEL message file which is being wrapped by the PLL message;

The *PLLPSICMiddleMDFile* consists of an MPEG-21 DID with a structure consisting of the following:

- an *Item* element (identified as *pllSensitiveInfo*), which carries the PLL specific metadata of the PLL message;
- any number of *Item* elements carrying the remaining PLL level parameters of $Info_S$.

### 7.3.2 Peer Registration Certificate

Upon a successful completion of its registration with the system, a peripheral or outer core, peer, $P_i$, is given a Registration Certificate by the *CCP*. In section 5.4.3 it is logically described as $rcertif_{P_i} = \text{signed}_{K_{CCP}^{-1}}\left(id \| SigInfo \| sid_{P_i} \| peerType \| vterm\right)$, where $id$ is the certificate's identifier, $SigInfo$ contains the necessary data to enable the validation of the certificate's signature, $peerType$ is the structural type of the peer (peripheral or outer core or central core), $sid_{P_i} = \text{signed}_{K_{P_i}^{-1}}\left(id_{P_i}\right)$ and $id_{P_i} = \left(p_i \| AlgrthmInfo \| k_{P_i} \| k_{P_i}^{-1} \| K_{P_i}\right)$. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 47.

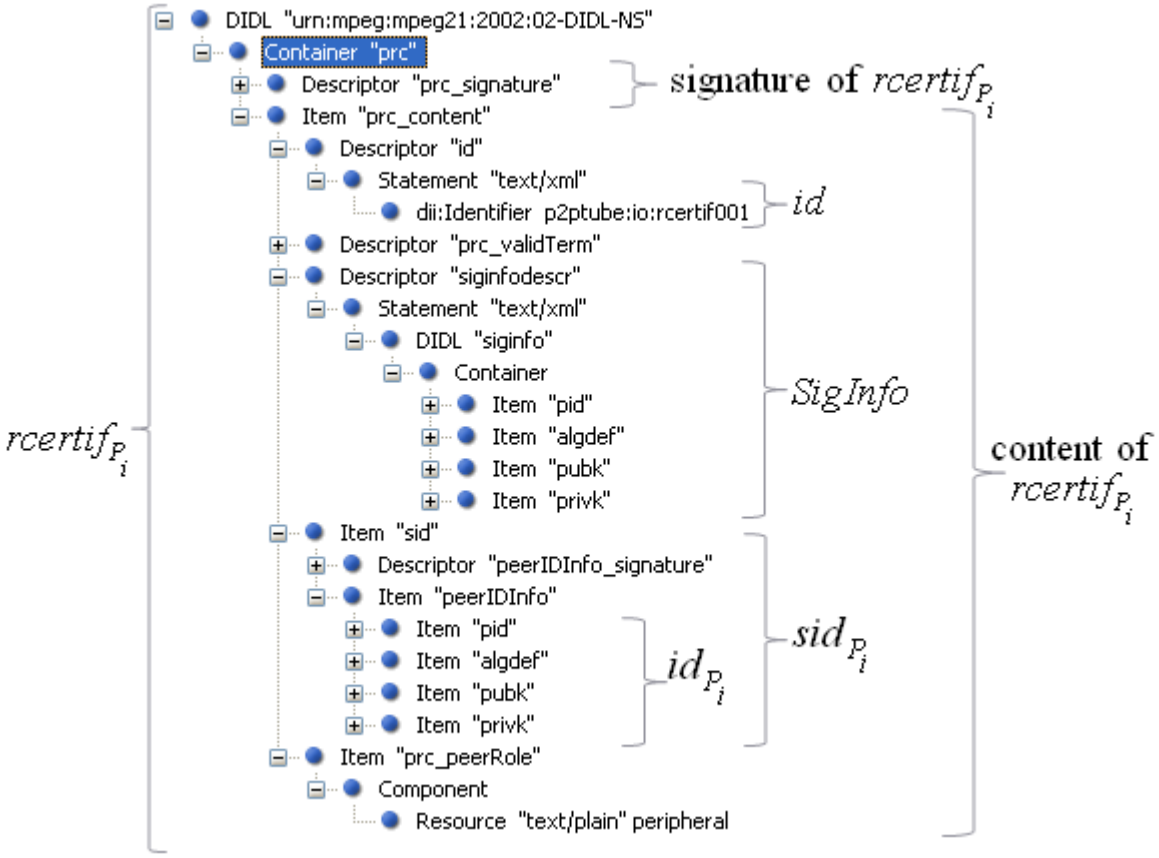


Figure 47 – Peer Registration Certificate Structure

Said structure, (for a hypothetical peripheral peer $P_i$), consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the certificate. This latter element carries:
  - a *Descriptor* element containing *CCP*'s signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element (identified as *prc_content*), which carries the actual contents of the certificate. It contains:
    - a *Descriptor* containing the certificate's identifier;

- a *Descriptor* containing the specification of the certificate's validity term.
- A *Descriptor* (identified as *siginfodescr*) which consists of the $SigInfo$ parameter;
- an *Item* (identified as *sid*) which consists of the $sid_{P_i}$ parameter;
- an *Item* containing the specification of the role played by $P_i$ in the system (either *peripheral*, *outercore* or *centralcore*);

The $SigInfo$ *Descriptor* contains (inside its *Statement* child) a root *DIDL* element carrying a *Container* element which functions as the top element of the rest of $SigInfo$'s structure. This latter element carries:

- an *Item* containing the identification of the siging peer ($P_i$);
- an *Item* containing the employed siging algorithm.
- an *Item* containing $P_i$'s public key and its identifier;
- an *Item* containing $P_i$'s private key identifier;

The $sid_{P_i}$ *Item* contains:

- a *Descriptor* element containing $P_i$'s signature of the remaining content of its parent *Item* element (the signature of its *Item* sibling);
- an *Item* which consists of the $id_{P_i}$. It contains:
  - an *Item* containing $P_i$'s id;
  - an *Item* containing $P_i$'s siging algorithm.
  - an *Item* containing $P_i$'s public key and its identifier;
  - an *Item* containing $P_i$'s private key identifier;

An example of a Peer Registration Certificate may be found in section A.2.2 of Annex A.

### *7.3.3 Peer Connection Certificate*

Upon a successful establishing of a connection between two peers ($P_i$ and $P_k$), the "client" peer (the peer which requested the establishment of the connection e.g. $P_i$), is given a Connection Certificate by the "server" peer (the peer to whom the establishment of the connection was requested e.g. $P_k$).

In section 5.4.5 said certificate is logically described as $ccertif_{P_i}^{P_k} = \text{signed}_{K_{P_k}^{-1}}\left(\left(id \| SigInfo \| rcertif_{P_i} \| rcertif_{P_k} \| vterm \| s_g^{P_i - CCP} \| enc_{K_{P_i}}\left(K_s^{s_g^{P_i - CCP}}\right)\right)\right)$. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 48.

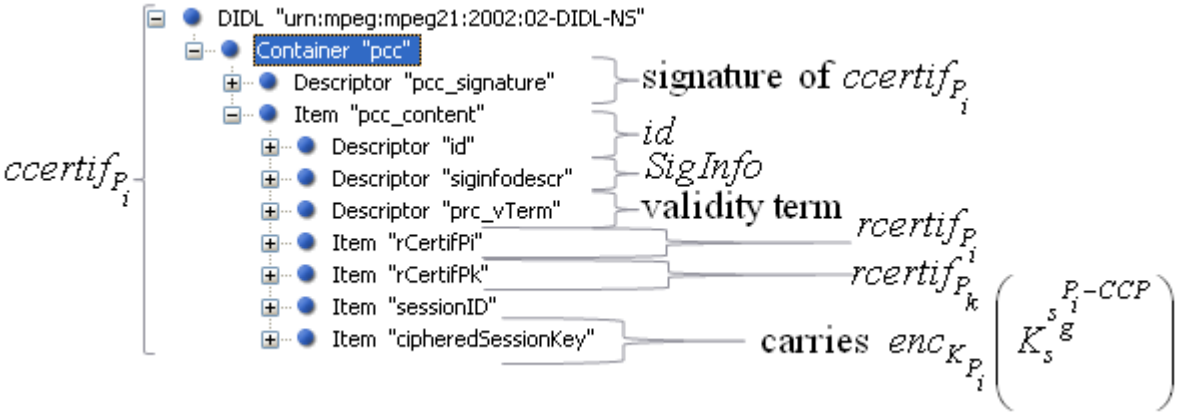


Figure 48 – Peer Connection Certificate Structure

Said structure, (for a hypothetical peripheral peer $P_i$ initiating a communication session with peer $P_k$), consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the certificate. This latter element carries:a *Descriptor* element containing the signature (by $P_k$), of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
- an *Item* element (identified as *pcc_content*), which carries the actual contents of the certificate. It contains:
  - a *Descriptor* containing the certificate's identifier;
  - a *Descriptor* containing the $SigInfo$ parameter;
  - a *Descriptor* containing the validity term of the connection certificate;
  - an *Item* (identified as *rCertifPi*) which contains $rcertif_{P_i}$;
  - an *Item* (identified as *rCertifPk*) which contains $rcertif_{P_i}$;
  - an *Item* (identified as *sessionID*) which contains $s_g^{P_i-CCP}$;
  - an *Item* (identified as *cipheredSessionKey*) which contains $enc_{K_{P_i}}\left(K_s \; s_g^{P_i-CCP}\right)$;

### *7.3.4 Peer Info Object*

In section 5.4.7 the Peer Info Object, of a specific peer $P_i$, built by a specific $OCP_j$, is logically described as $pInfo_s = \text{signed}_{K^{-1}_{OCP_j}}\left(id \| SigInfo \| rcertif_{P_i} \| IP_i \| vterm\right)$. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 49.

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the object. This latter element carries:
  - a *Descriptor* element containing an $OCP$ 's signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element (identified as *pio_content*), which carries the actual contents of the object. It contains:
    - a *Descriptor* containing the object's identifier;

- a *Descriptor* element containing the object's validity term;
- a *Descriptor* element containing the $SigInfo$ parameter;
- an *Item* containing $P_i$'s IP address;
- an *Item* containing $P_i$'s registration certificate.

The last mentioned *Item* carries a *Component* which carries a *Resource* element. The latter contains, inline, $P_i$'s registration certificate. The structure of this certificate is the one presented in section 7.3.2.

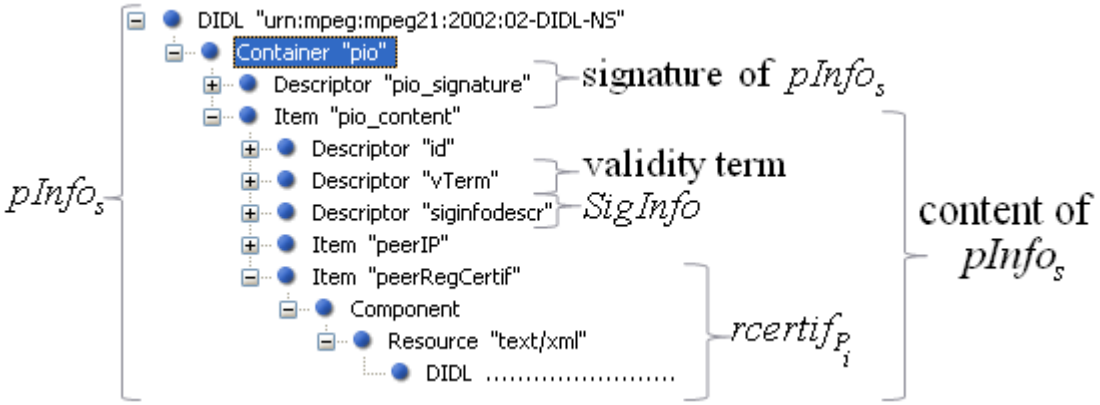


Figure 49 – Peer Info Object Structure

An example of a Peer Info Object may be found in section A.2.3 of Annex A.

### *7.3.5 Peer Quarantine and Expulsion Lists*

In section 5.4.7 the peer quarantine list (compiled by $OCP_j$), is logically defined as $quarList_s = \text{signed}_{K^{-1}_{OCP_j}}\left(SigInfo \| QList \| tstamp\right)$. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 50.

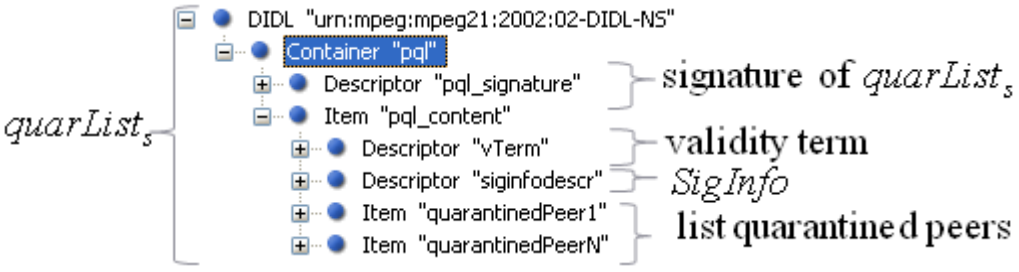


Figure 50 – Peer Quarantine List Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the object. This latter element carries:
  - a *Descriptor* element containing an $OCP$'s signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element (identified as *pql_content*), which carries the actual contents of the object. It contains:
    - a *Descriptor* element containing the object's validity term;
    - a *Descriptor* element containing the $SigInfo$ parameter;

- any number of *Item* elements, each containing the identifier of a quarantined peer;

The structure of a peer expulsion list, logically defined as $expList_s = \text{signed}_{K^{-1}_{OCP_j}}\left(SigInfo \| XList \| tstamp\right)$, in section 5.4.7, is the same as that of the peer quarantine list.
An example of a Peer Quarantine List may be found in section A.2.4 of Annex A.

### *7.3.6 PLL Info Retrieval Permit*

In section 5.4.7 the Peer Info Retrieval Permit, (a permit, issued by a specific $OCP_j$, enabling a specific peripheral peer $PP_i$ to retrieve some PLL level information from another peripheral peer, $PP_j$ ), is logically described as:

$$retrpermit^{PP_i}_{pllInfo} = \text{signed}_{K^{-1}_{OCP_j}}\left(\text{id} \| \text{pllInfoRetrPerm} \| SigInfo \| pp_i \| vterm \| infoDef \| id_{pllildo_s}\right),$$

where. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 51.

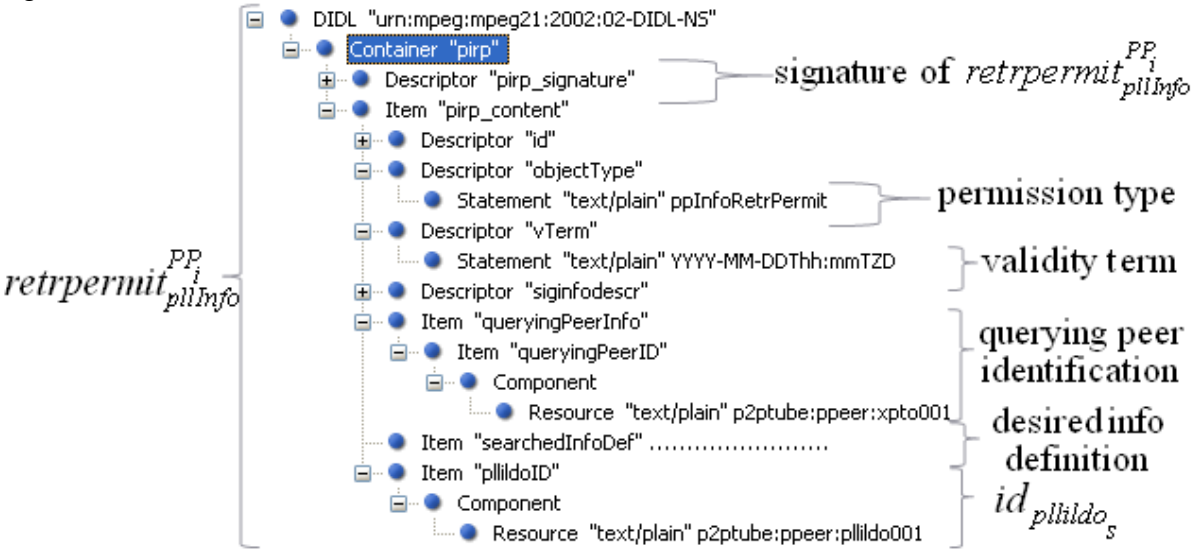


Figure 51 – Peer Info Retrieval Permit Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the permit. This latter element carries:
  - a *Descriptor* element containing an *OCP* 's signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element (identified as *piorp_content*), which carries the actual contents of the permit. It contains:
    - a *Descriptor* element containing the permit's identifier;
    - a *Descriptor* element containing the designation of the type of the object (PPL Info Retrieval permit);
    - a *Descriptor* element containing the permit's validity term;
    - a *Descriptor* element containing the *SigInfo* parameter;

- an *Item* containing the relevant information about the peer to whom the permit has been issued ( $PP_i$ ). It contains an inner *Item* which carries (inside a *Resource* inside a *Component*), $PP_i$ 's identifier;
- an *Item* containing the information which $PP_i$ is allowed to retrieve. It corresponds to the *infoDef* parameter;
- an *Item* containing the identifier of the permit's corresponding $id_{pllildo_s}$ ;

An *exa*mple of a Peer Info Object Retrieval Permit may be found in section A.2.5 of Annex A.

### *7.3.7 PLL Information Location Describing Object*

In section 5.4.7 the PLL Information Location Describing Object, is logically described as:

$$pllildo_s = \text{signed}_{K^{-1}_{OCP_j}}\left( id \| \ SigInfo \| infoDef \| vterm \| \left( rcertif_{PP_i} \| ..... \| rcertif_{PP_n} \right) \right).$$

In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 51.

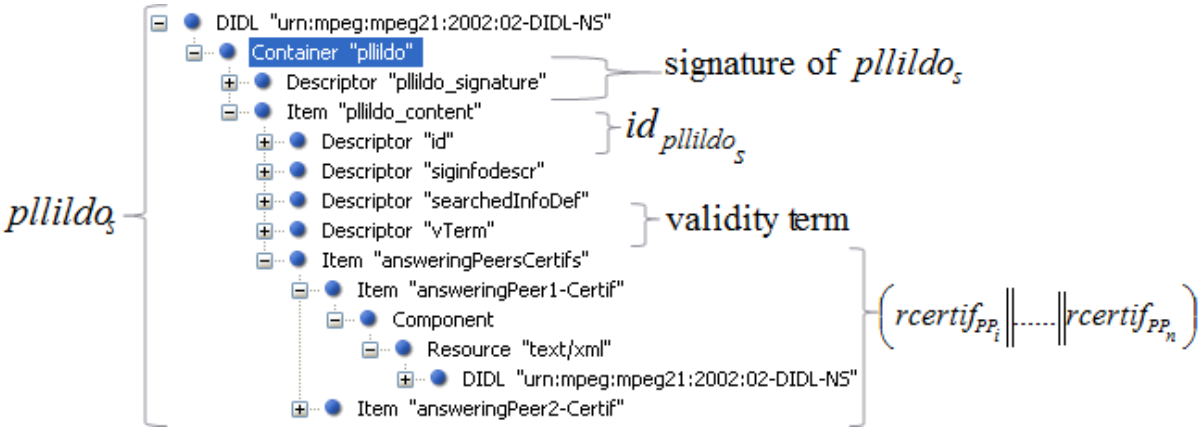


Figure 52 – PLL Information Location Describing Object

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the object. This latter element carries:
  - a *Descriptor* element containing an *OCP* 's signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element (identified as *pllildo_content*), which carries the actual contents of the object. It contains:
    - a *Descriptor* element containing the object's identifier;
    - a *Descriptor* element containing the *SigInfo* parameter;
    - a *Descriptor* element containing the information which is located by this object. It corresponds to the *infoDef* parameter;
    - a *Descriptor* element containing the object's validity term;
    - an *Item* containing the necessary registration certificates of the peers which hold the information defined in the *infoDef* parameter;

## 7.4 UEL Objects

### *7.4.1 UEL Message*

In section 5.5.2 an UEL message (whose production involves user $U_i$) is, logically defined (in an implicit manner) as $\left( \text{signed}_{K_{U_i}^{-1}} \left( SigInfo \| serial \| UELinfo_s \right) \| UELinfo \right)$.

In more precise terms they consist of a TAR archive with the following content:

- UEL Sensitive Info File (*UELSIFile*) – contains the UEL message's sensitive information.
- None or one UEL Non Sensitive Info File (*UELNSIFile*) – carries non sensitive UEL information. It consists of a TAR archive containing:

The *UELSIFile* (which basically contains the $\text{signed}_{K_{U_i}^{-1}} \left( SigInfo \| serial \| UELinfo_s \right)$ part), consists of an MPEG-21 DID with the structure depicted in Figure 53.

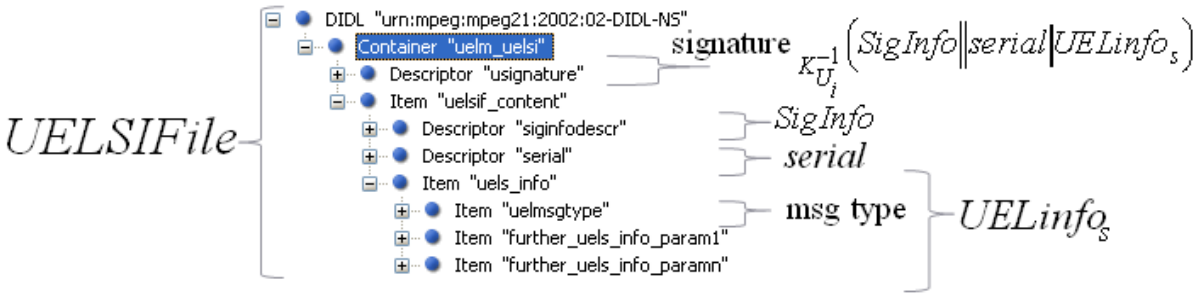


Figure 53 – UELSIFile Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the file's contents. This latter element carries:
  - a *Descriptor* element containing the user's signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element containing:
    - a *Descriptor* element containing the $SigInfo$ parameter;
    - a *Descriptor* element containing the serial of the UEL message;
    - an *Item* element (identified as *uels_info*), which carries the sensitive UEL information of the message;
    - an *Item* element (identified as *uelm_uelsi_ssinfo*), which carries the sensitive UEL information of the message. It contains:
      - an *Item* element carrying the definition of the type of UEL message;
      - zero or more *Item* elements carrying futher sensitive UEL parameters;

An example of the contents of a *UELSIFile* may be found in section A.3.1.1 of Annex A.

### *7.4.2 User Registration Certificate*

After a user, $U_i$, successfully finishes his registration with the system, he is given a User Participation Certificate. In section 5.5.3 it is logically described by the following equation $rcertif_{U_i} = \text{signed}_{K_{CCP}^{-1}}\left(id \| SigInfo \| sid_{U_i} \| vterm\right)$. The definitions of some of the most relevant of its parameters are $sid_{U_i} = \text{signed}_{K_{U_i}^{-1}}\left(id_{U_i}\right)$ and $id_{u_i} = \left(u_i \| uname_{U_i} \| AlgrthmInfo \| k_{U_i} \| k_{U_i}^{-1} \| K_{U_i}\right)$. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 54.

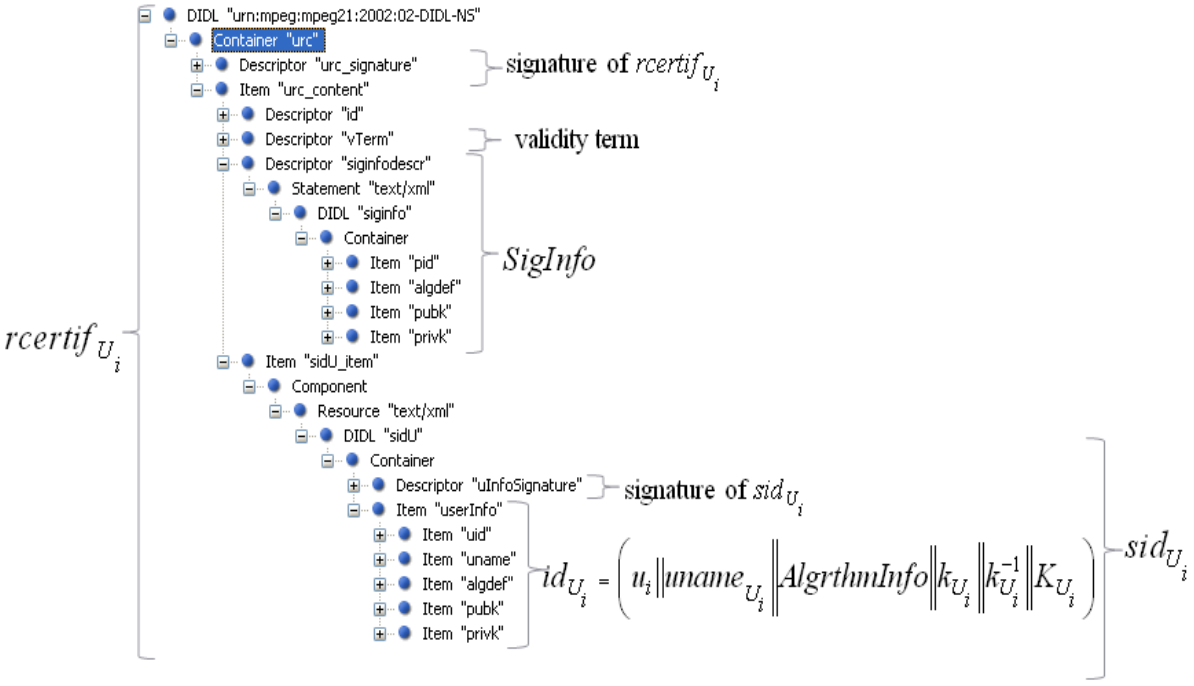


Figure 54 – User Registration Certificate Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the certificate. This latter element carries:
  - a *Descriptor* element containing the $CCP$'s signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element which carries the actual content of the certificate. It contains:
    - a *Descriptor* containing the certificate's identifier;
    - a *Descriptor* element containing the validity term of the certificate;
    - a *Descriptor* element containing the $SigInfo$ parameter;
    - an *Item* element which carries the $sid_{U_i}$. It contains a DID which carries the actual $sid_{U_i}$. Said DID has at its root a *DIDL* element carrying a *Container* element which contains:
      - a *Descriptor* element containing $U_i$'s signature of the remaining content of its parent *Item* element (the signature of its *Item* sibling);

- an *Item* which constitutes $id_{U_i}$. It contains five inner *Items* which respectively carry:
  - o $U_i$'s identifier;
  - o $U_i$'s username;
  - o $U_i$'s signing algorithm definition;
  - o $U_i$'s public key and its identifier;
  - o The identifier of $U_i$'s private key;

An example of a User Registration Certificate may be found in section A.3.2 of Annex A.

### *7.4.3 User Hosting Certificate*

After a user, $U_i$, successfully logs into the system, his hosting peripheral peer, $PP_i$, his given, by $CCP$, a certificate proving that $PP_i$ is in fact hosting user $U_i$. In section 5.5.5 said certificate is logically described as $hostcertif_{U_i}^{PP_i} = \text{signed}_{K_{CCP}^{-1}}\left(id \| SigInfo \| rcertif_{U_i} \| pp_i \| vterm\right)$, where $rcertif_{U_i} = \text{signed}_{K_{CCP}^{-1}}\left(id \| SigInfo \| sid_{U_i} \| vterm\right)$. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 55.

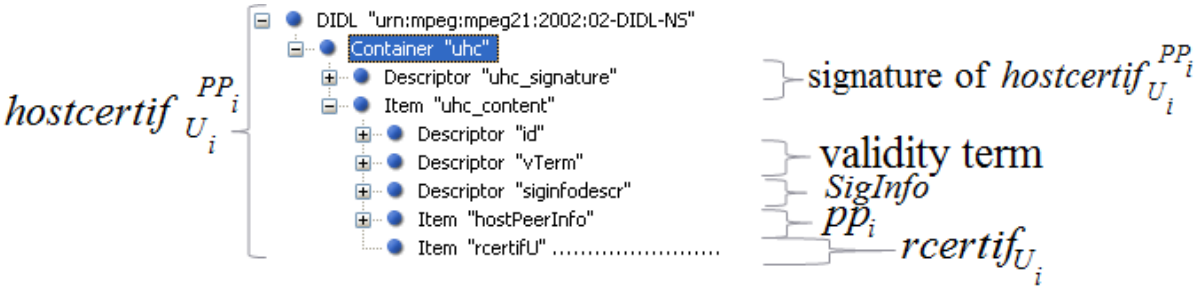


Figure 55 – User Hosting Certificate Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the certificate. This latter element carries:
  - o a *Descriptor* element containing the $CCP$'s signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - o an *Item* element which contains:
    - a *Descriptor* containing the certificate's identifier;
    - a *Descriptor* element containing a timestamp with the certificate's validity term;
    - a *Descriptor* element containing the $SigInfo$ parameter;
    - an *Item* which contains the identification of the hosting peer;
    - an *Item* which constitutes the $rcertif_{U_i}$. Its contents and structure are the same as those explained in section 7.4.2.

An example of a User Hosting Certificate may be found in section A.3.3 of Annex A.

### *7.4.4 Search Query Response Objects*

Before a user decides to consume a specific Media Object (MO), he must first learn about the existence of such an object. That knowledge is typically obtained through content searches (e.g. googling).

The user will thus specify a number of semantic criteria that his target MO(s) must respect and tell his hosting peer, $PP_i$, to obtain a list with such MOs. The searching procedure that will then ensue (which will unfold in the manner presented in section 5.5.8.2.2), will ultimately result in the reception, by $PP_i$, of a Search Query Response Object ($sqro$), (sent from the $CCP$, from an $OCP$ or from another peripheral peer), originally produced by the *CCP*, or an $OCP$, which contains the response. An $sqro$ may be logically described as $sqro = \text{signed}_{K_{CCP}^{-1}}\left(id \| SigInfo \| tstatmp \| answeredQ \| moList\right)$ (alternatively it may instead be signed by an $OCP$), where $answeredQ = \left(\text{signed}_{K_{CCP}^{-1}}\left(id \| SigInfo \| query\right)\right)$, and $moList = \left(\left(mo_1 \| semchars_{MO_1}\right) \| \left(\text{.....}\right) \| \left(mo_n \| semchars_{MO_n}\right)\right)$. In more precise terms it consists of an MPEG-21 DID with the structure depicted in Figure 56.

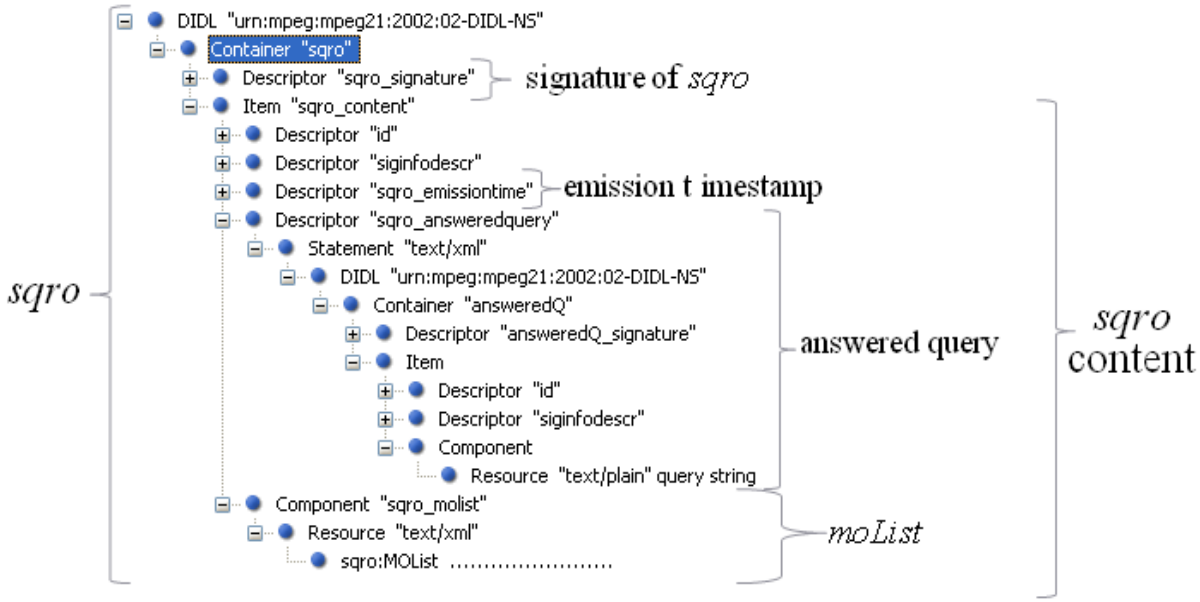


Figure 56 – SQRO Structure

Said structure consists of the following:

- an inner *Item* – it carries:
  - the identifier of the object – within Descriptor "*id*";
  - the *SigInfo* parameter – within *Descriptor* "*siginfodescr*";
  - the emission date of the query response information – within *Descriptor* "*sqro_emissiontime*";
  - the answered query – within *Descriptor* "*sqro_answeredquery*": It carries an inner DID which constitutes the answered query object. It contains:
    - a *Descriptor* carrying the signature of the answered query object's content;
    - an *Item* carrying the actual content of the object. It contains:

- a *Descriptor* carrying the objects identifier;
- a *Descriptor* carrying the adequate $SigInfo$ parameter;
- a *Component* carrying, inline, the actual query string;

  - the actual query response information – within the "*sqro:MIList*" child of its *Component* child;
- an outer *Container* – it carries:
  - the inner *Item*;
  - the security assuring provisions – consist of the digital signature, (by $CCP$ or an $OCP$ ), of the inner *Item*, expressed by a *dsig:Signature* element of *Descriptor* "*sqro_signature*".

The *sqro:MOList* element is structured in accordance with the schema depicted in Figure 57.

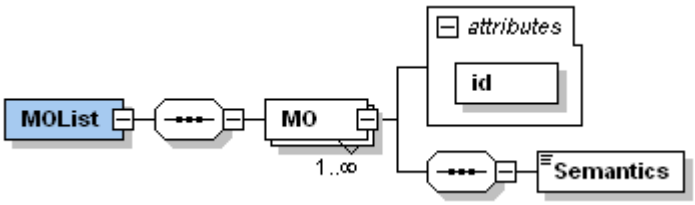


Figure 57 – MOList Schema Depiction

The signing of the inner item, (by the $CCP$ or by an $OCP$ , at $sqro$ 's original production time), assures $sqro$ 's integrity and authenticity during its propagation across the system's tissue, thus enabling its P2P diffusion, independently of the $CCP$ or of the $OCPs$ . *SQRO*'s emission date, contained within the "*sqro_emissiontime*" *Descriptor*, enables any peripheral peer to assess the freshness or staleness of the information.
The signing of the "answered query object" assures that it has the same security and distribution related properties, as the contents of the above mentioned inner Item.

An example of an $sqro$ may be found in section A.3.4 of Annex A.

### *7.4.5 UEL Information Location Describing Objects*

The UEL Information Location Describing Object contains the necessary information for a peer to know where to retrieve a specific portion of UEL information (which may be an MO, an $sqro$ , etc.), from. It may be described by the following equation,

$uelildo_s = \text{signed}_{K^{-1}_{OCP_i}}\left(id \| SigInfo_{OCP_i} \| tstamp \| loclist \| specificinfo\right)$, where $id$ is the object's identifier, $SigInfo_{OCP_i}$ carries the necessary information to validate $OCP_i$'s signature of $uelildo_s$, $tstamp$ is $uelildo_s$'s emission time, $loclist = \left(PP_1 \| ..... \| PP_n\right)$ is the list of peers from where the target information object may be retrieved, and $specificinfo$ is an optional parameter whose presence and contents depend on the type of the target UEL information.

In section 5.5.8.2.2, an example of an MO retrieval operation is given. In that specific case, the employed $uelildo_s$ takes the form

$uelildo_s = \text{signed}_{K^{-1}_{OCP}}\left(id \| SigInfo_{OCP_i} \| tstamp \| loclist \| fraglist\right)$, where

$$fraglist = \text{signed}_{K_{CCP}^{-1}}\left( id \left\| SigInfo \right\| mo_i^{CCP} \left\| \left( fragdef_0^{MO_i^{CCP}} \right\| ..... \right\| fragdef_n^{MO_i^{CCP}} \right)\right), \text{ and}$$

$$fragdef_j^{MO_i^{CCP}} = \left( frag_j^{MO_i^{CCP}} \left\| size \right\| seqnr \left\| \left( SigInfo \left\| signature_{K_{CCP}^{-1}} \left( Frag_j^{MO_i^{CCP}} \right) \right) \right) \right).$$

That is, $specificinfo = fraglist$. $fraglist$ is the list of fragments into which the media object, $MO_i^{CCP}$, is divided into. The rest of this section will use this specific instance of a $uelildo_s$ to exemplify it.

In precise terms, a $uelildo_s$ consists of an MPEG-21 DID with the structure depicted in Figure 58.

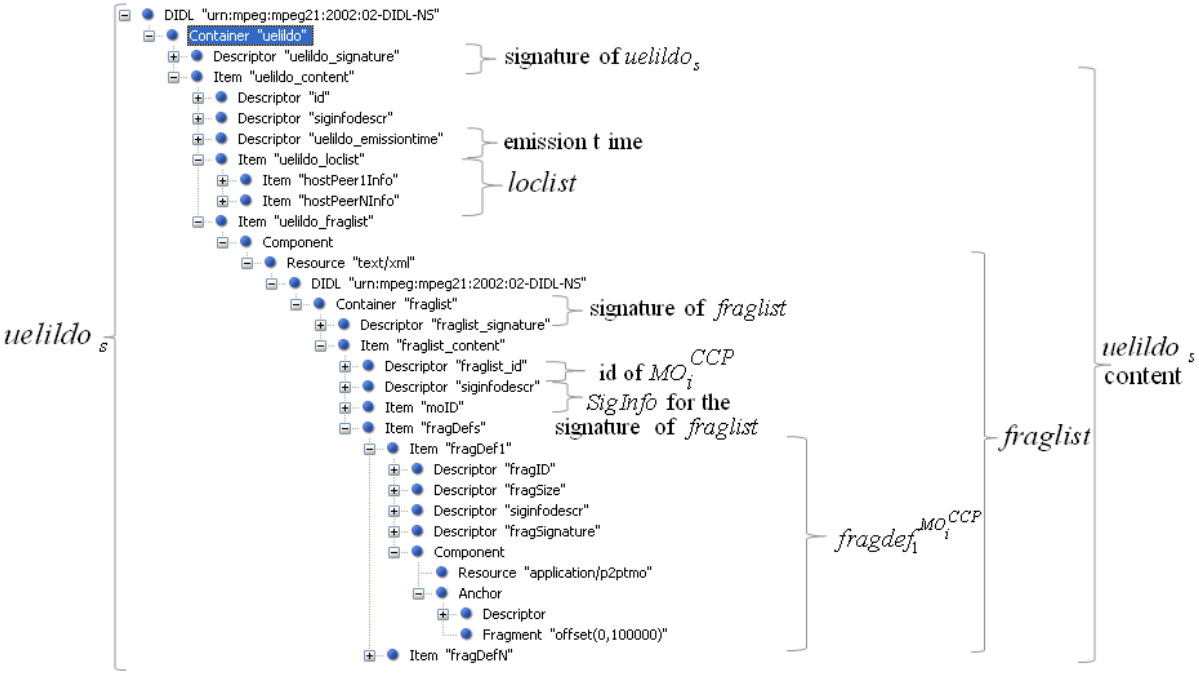


Figure 58 – UELILDO Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the object. This latter element carries:
  - a *Descriptor* element containing the $CCP$'s (or an *OCP*'s) signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* – it contains:
    - a *Descriptor* containing the object's identifier;
    - a *Descriptor* element containing the $SigInfo$ parameter, with the necessary data to enable the validation of the $uelildo_s$ signature;
    - a *Descriptor* element containing the IOLDO's emission timestamp;
    - an *Item* containing the $loclist$. It contains:
      - an *Item* element, for each of the peers hosting the MO. Each such *Item* contains the identification of a specific peer;
    - an *Item* containing the $fraglist$. It contains the $fraglist$ carrying DID. It thus carries (inside the *Resource* child of a *Component* child):

- A top *DIDL* element carrying a *Container* element which functions as the top element of the $fraglist$. This latter element carries:
- a *Descriptor* containing $fraglist$'s signature;
- an *Item* carrying $fraglist$'s actual content. It contains:
    - a *Descriptor* containing the object's identifier;
    - a *Descriptor* element containing the $SigInfo$ parameter, with the necessary data to enable the validation of $fraglist$'s signature;
    - an *Item* carrying the identification of the MO at stake;
    - an *Item* element, for each of the $Fragdef$ objects describing MO fragments. Each such *Item* contains:
        - an *Descriptor* element carrying the fragment ID;
        - an *Descriptor* element specifying the fragment size;
        - a *Descriptor* element containing the $SigInfo$ parameter, with the necessary data to enable the validation of fragment's signature (bellow);
        - an *Item* element carrying the fragment's signature;
        - a *Component* element which defines the fragment at stake.

Said *Components* carry an *Anchor* element, which carries the identification of the fragment (within a *Descriptor*), and carries the fragment's definition within a *Fragment* element [175].

The signing of Item "*uelildo_content*" by $CCP$ (or by some *OCP*) enables the validation of $uelildo_s$'s integrity and of its origin authenticity.

Besides the description of the location of MOs, $uelildo_s$ may also be used, for instance, for the description of the location of $rcertif_{U_i}$, $hostcertif_{U_i}^{PP_i}$, or other $uelildo_s$. In all such cases $specificinfo = \emptyset$.

An example of a UELILDO, of the above exemplified type, may be found in section A.3.5 of Annex A.

### *7.4.6 UEL Info Retrieval Permit*

A UEL Info Retrieval Permit is a permit enabling a specific peripheral peer $PP_i$, to retrieve a specific piece of UEL information, from a number of other peers. It may logically be defined as

$$retrpermit_{uelInfo}^{PP_i} = \text{signed}_{K_{OCP_i}^{-1}}\left(\text{id}\left\|\text{uelInfoRetrPerm}\right\|SigInfo_{OCP_i}\left\|pp_i\right\|vterm\left\|infoDef\right\|id_{uelildo_s}\right).$$

In section 5.5.8.2.1.1 such a permit is employed to enable the retrieval of an MO (e.g. $MO_i^{CCP}$). In the remainder of this section we shall exemplify the structure of the permit in scope, employing a permit of the latter type. Thus, in this particular case, $infoDef = \left(mo_i^{CCP}\right)$ is the identifier of the desired MO.

In precise terms a UEL Info Retrieval Permit consists of an MPEG-21 DID with the structure depicted in Figure 59.

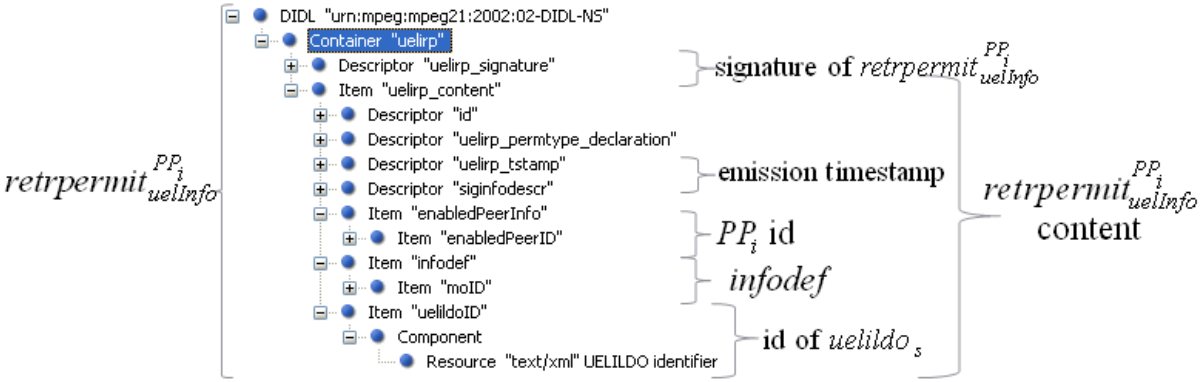


Figure 59 – MO Retrieval Permit Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the permit. This latter element carries:
  - a *Descriptor* element containing $CCP$'s, or an $OCP$'s, signature of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element (identified as *uelrip_content*), which carries the actual contents of the permit. It contains:
    - a *Descriptor* containing the certificate's identifier;
    - a *Descriptor* element containing the definition of the certificate's type;
    - a *Descriptor* element containing the certificate's emission timestamp;
    - a *Descriptor* containing the $SigInfo$ parameter;
    - an *Item* containing the identification of the peer to which the permit is granted;
    - an *Item* containing the definition of the information to which the peer is being granted access to;
    - an *Item* containing the identifier of the $uelildo_s$ which specifies where the targeted information may be retrieved from;

Besides the given example, this certificate may also be employed to enable a peer to retrieve other UEL information objects such as user registration certificates ($rcertif_{U_i}$), user hosting certificates, ($hostcertif^{PP_i}_{U_i}$) or content search response objects (see *SQRO* in section 7.4.4).

An example of an UEL Info Retrieval Permit may be found in section A.3.6 of Annex A.

### *7.4.7 Media Objects*

P2PTube's MOs consist of TAR archives which have the following content:

- MO Top Head File (*MOTHFile*) – this file contains all of the $CCP$ inserted metadata pertaining to the overall MO;
- MO Inner File (*MOIFile*) – this is a TAR archive that basically consists of the user delivered MO object at insertion time. It contains:

- MO Inner Head File (*MOIHFile*) – this file contains all the metadata of the *MOIFile*;
- One or more MO Inner Content File(s) (*MOICFile*) – each containing an actual media (video) content;

The contents of the *MOTHFile* may be defined as $MOTH_i^{CCP} = \left( \text{signed}_{K_{CCP}^{-1}} \left( mo_i^{CCP} \| SigInfo \| tstamp \| \text{signature}_{K_{CCP}^{-1}} (MOI_i) \right) \right)$, where $MOI_i$ corresponds to the *MOIFile*. It may be described as $MOI_i = \left( MOIH_i \| MOIC_1 \| ..... \| MOIC_n \right)$, where $MOIH_i$ corresponds to the *MOIHFile* and each $MOIC_i$ corresponds to a *MOICFile*. The earlier may be defined as $MOIH_i = \left( \text{signed}_{K_{U_i}^{-1}} \left( SigInfo \| semantics \| rights \| x \right) \right)$, where the latter variable may be defined as $x = \left( \left( moic_1 \| \text{signature}_{K_{U_i}^{-1}} (MOIC_1) \right) \| .... \| \left( moic_n \| \text{signature}_{K_{U_i}^{-1}} (MOIC_n) \right) \right)$ and $moic_i$ is the identifier of $MOIC_i$.

In more precise terms, a *MOTHFile* consists of an MPEG-21 DID with the structure depicted in Figure 60.

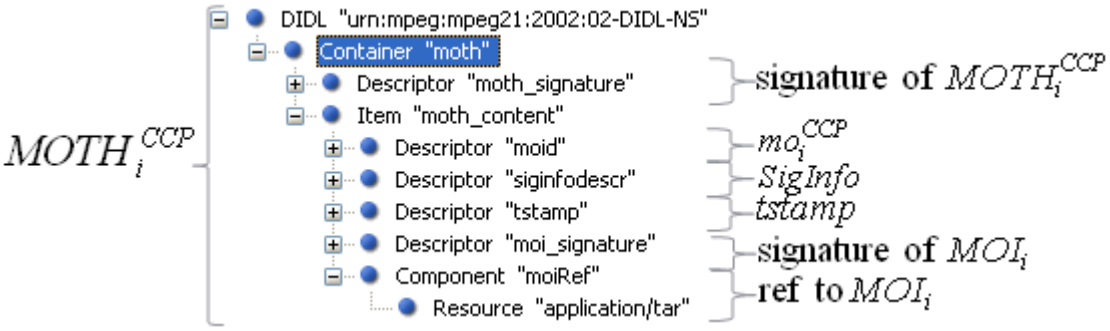


Figure 60 – MOTHFile Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the object. This latter element carries:
  - a *Descriptor* carrying the digital signature, (by $CCP$), of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an inner *Item* – it contains:
    - a *Descriptor* element carrying the identifier of the MO;
    - a *Descriptor* element carrying the $SigInfo$ parameter;
    - a *Descriptor* element carrying the time stamp indicating the MO's injection time, into the system;
    - a *Descriptor* element carrying the signature of the *MOIFile* TAR archive;
    - a *Component* element carrying a reference to the *MOIFile* TAR archive;

The signing, by $CCP$ , of the *MOTHFile*'s contents and of the *MOIFile* enables any peer in the system to verify the integrity of those files as well as the veracity of its acceptance by the system.

An example of a *MOTHFile*'s contents may be found in section A.3.7 of Annex A.

A *MOIHFile* consists of an MPEG-21 DID with the structure depicted in Figure 61.

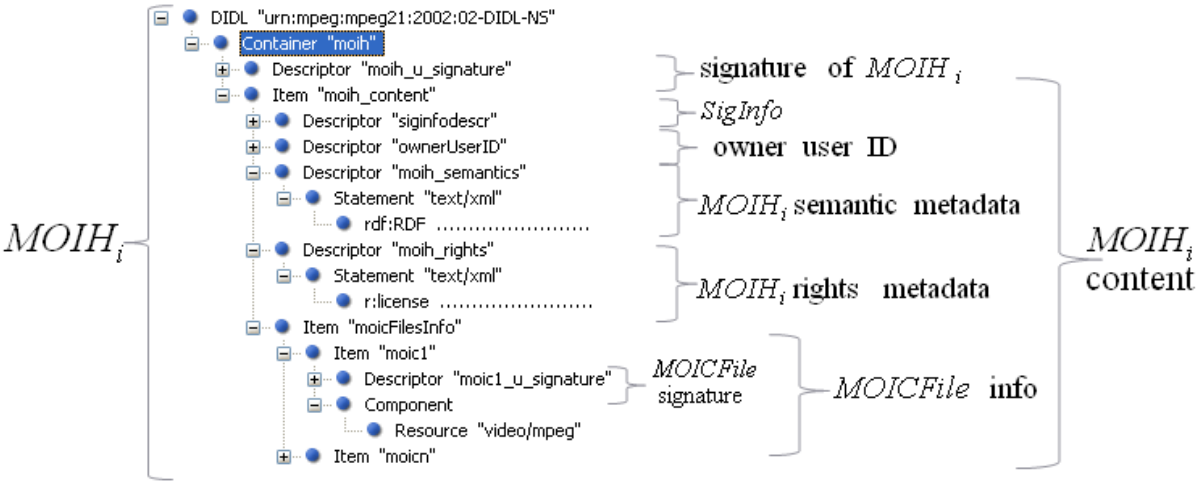


Figure 61 – MOIHFILE Structure

Said structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the object. This latter element carries:
  - a *Descriptor* carrying the digital signature, (by the MO's issuing user, e.g. $U_i$ ), of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an *Item* element – it contains:
    - a *Descriptor* element carrying the $SigInfo$ parameter;
    - a *Descriptor* element (identified as *moih_ownerU*) carrying the identity of the MO's owner user ($u_i$ );
    - a *Descriptor* element (identified as *moih_semantics*) carrying the MO's semantically describing metadata. This metadata describes the semantic characteristics of the MO and its relationships with other MOs. Said relationships are described employing MPEG-21's relationship describing capabilities which were added to that work as a result of this PhD work (see section 4.3 of chapter VI). Semantic information is typically is expressed in the RDF format;
    - a *Descriptor* element (identified as *moih_rights*), carrying the rights information pertaining to the MO. This information is carried by an REL element rel:license [113]. In light of the nature of the business models that this architecture is meant to enforce (presented in chapter IV), and of its implications on MO access by users, the purpose of this rights information is merely to formalize the openness of content access;
    - an *Item* element carrying the information pertaining to the MO's *MOICFiles*: It contains:
      - one or more *Item* elements. Each such element carries the information pertaining to a single *MOICFile*, and, thus, contains:

- a *Descriptor* (identified as *moic1_u_signature*), carrying the digital signature, (by $U_i$), of the corresponding *MOICFile*;
- a *Component* carrying a reference to the *MOICFile* in scope;

The signing, by $U_i$, of each of the MO's *MOICFiles* enables the validation of their integrity and of their origin authenticity. It also guarantees their non repudiability. Furthermore, the signing, by $U_i$, of the *moih_content Item* enables the same actions towards that *Item*, thus securing the semantic and rights metadata.

In accordance with what was expressed in section 5.5.8.2.1.1, at the production time of an MO, $MO_i$, $U_i$ produces the signatures of all *MOICFiles*, prepares the *moih_content Item* declaration, produces the signature of the latter *Item* and then finishes the production of the *MOIHFile* and of the *MOIFile*.

This way $U_i$ produces an $MO_i$ which is effectively signed by him. $MO_i$ is sent to $CCP$ which calculates its own signature of the *MOIFile*, builds the *MOTHFile*, and then repackages the set (*MOTHFile* and *MOIFile*) into what then is its distribution ready form, $MO_i^{CCP}$.

An example of a *MOIHFile*'s contents may be found in section A.3.7 of Annex A.

### *7.4.8 MO Ransom Announcement*

An MO Ransom Announcement may be defined as $RnsmAnn^{MO_i^{CCP}} = \left(id \| mo_i^{CCP} \| u_i \| ammount \| semantics\right)$. It thus contains its own identifier, the identifier of the prospective MO, the identifier of the MOs producing user (which is to be the one to be donated to), the specification of the required ransom amount and the semantic characterization of the prospective MO.

### *7.4.9 User Monitoring Requests and Responses*

The ERR and ER objects mentioned in section 5.5.6 may logically be described, respectively as $err_{CCP}^{PP_i} = \text{signed}_{\text{K}_{CCP}^{-1}}\left(id \text{ of } err_{CCP}^{PP_i} \| SigInfo_A \| err \| PP_i\right)$, (where $SigInfo_A = \left(ccp \| AlgrthmInfo \| k_{CCP} \| k_{CCP}^{-1} \| K_{CCP}\right)$ and $err$ is the actual MPEG-21 Part 15, event report requesting information), and $er_{PP_i}^{CCP} = \text{signed}_{\text{K}_{PP_i}^{-1}}\left(SigInfo_B \| \left(\text{signed}_{\text{K}_{U_i}^{-1}}\left(id \text{ of } er_{PP_i}^{CCP} \| SigInfo_C \| id \text{ of } err_{CCP}^{PP_i} \| er\right)\right)\right)$ (where $er$ is the reported information, $SigInfo_B = \left(pp_i \| AlgrthmInfo \| k_{PP_i} \| k_{PP_i}^{-1} \| K_{PP_i}\right)$ and $SigInfo_C = \left(u_i \| AlgrthmInfo \| k_{U_i} \| k_{U_i}^{-1} \| K_{U_i}\right)$).

In more precise terms, they are expressed as MPEG-21 DIDs carrying MPEG-21 ERRs and MPEG-21 ERs respectively [102]. Figure 62 depicts the structure of a P2PTube ERR and Figure 63 depicts the structure of a P2PTube ER.

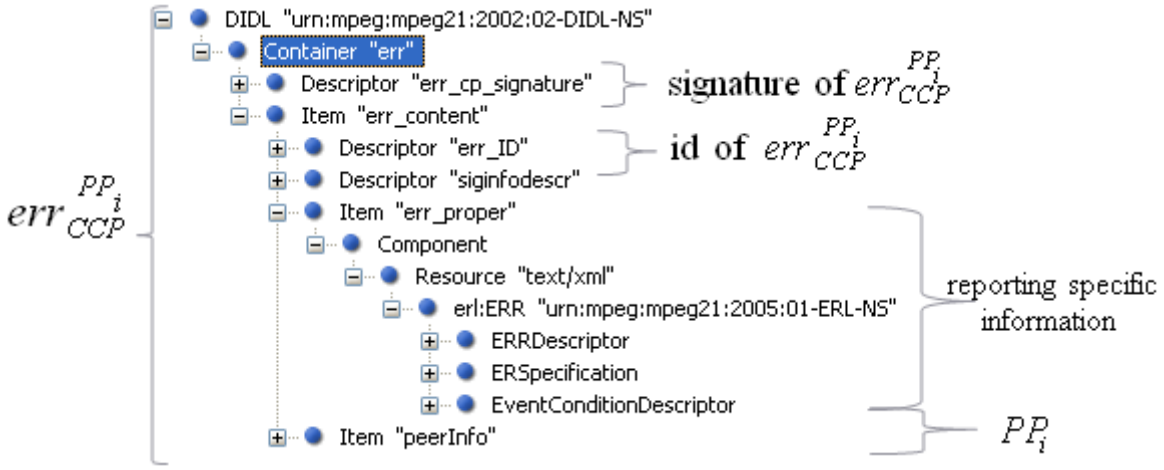


Figure 62 – ERR Structure

An ERR's structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the report. request This latter element carries:
  - a *Descriptor* carrying the digital signature, (by $ccp$), of the remaining content of its parent *Container* element (the signature of its *Item* sibling);
  - an inner *Item* – it contains:
    - a *Descriptor* carrying the identification of the ERR;
    - a *Descriptor* carrying the $SigInfo$ parameter;
    - an *Item* carrying the actual ERR information. It carries that information as an *erl:ERR* element [102];
    - an *Item* element carrying the identification of the event report request's target peer (e.g. $pp_i$);

The presence of *CCP*'s signature of *err_content Item*'s (inside *Descriptor err_cp_signature*), enables the peer receiving the ERR to validate its origin authenticity.

An example of a P2PTube's ER may be found in section A.3.8 of Annex A.

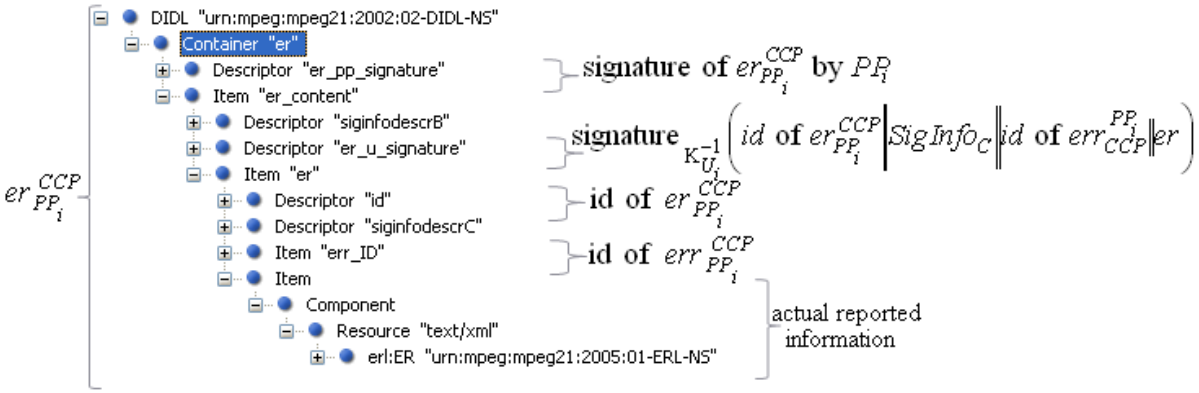


Figure 63 – ER Structure

An ER's structure consists of the following:

- A root *DIDL* element carrying a *Container* element which functions as the top element of the report. This latter element carries:
  - a *Descriptor* carrying the digital signature, (by $PP_i$), of its *Item* sibling;
  - an inner *Item* – it contains:
    - a *Descriptor* carrying the $SigInfo_B$ parameter;
    - a *Descriptor* carrying the digital signature, (by $U_i$), of its *Item* sibling;
    - an inner *Item* – it contains:
      - a *Descriptor* carrying the identification of the ER;
      - a *Descriptor* carrying the $SigInfo_C$ parameter;
      - an *Item* carrying the identifier of the original event eport request that caused the emission of this event repor;
      - an *Item* carrying the actual ER information as its inline resource. It carries that information as an *erl:ER* element [102];

The presence of $PP_i$'s signature of *er_content Item* (inside *Descriptors er_pp_signature*), enables the $CCP$ to check if the report was in fact sent by $PP_i$. The presence of $U_i$'s signature of the *er Item* (inside *Descriptor er_u_signature*), enables $CCP$ to determine if the involved user was in fact the targeted one.

An example of a P2PTube's ER may be found in section A.3.8 of Annex A.

### *7.4.10 Inquiry and Inquiry Response Objects*

An Inquiry IO is associated to a specific advertisement MO. It carries a set of multiple choice style queries pertaining the semantic content of the associated advertisement MO. An Inquiry Response IO carries the indication of the correct response alternatives to the queries in its corresponding Inquiry IO.

When an advertiser inserts an advertising MO he also supplies the associated Inquiry IOs and Inquiry Response IOs (potentially dozens of them).

An Inquiry IO may be described as $inquiryIO_i = \text{signed}_{K^{-1}_{U_x}}(SigInfo \| x)$ and

$$x = \left( id \left\| \left( query_1 \left\| \left( qRespAltern_1^1 \right\| .... \left\| qRespAltern_1^n \right) \right) \right\| .... \left\| \left( query_n \left\| \left( qRespAltern_n^1 \right\| .... \left\| qRespAltern_n^n \right) \right) \right) \right. \right.$$

An Inquiry Response IO may be described as $inquiryRespIO_i = \text{signed}_{K^{-1}_{U_x}}(SigInfo \| x)$ where

$x = \left( id \left\| \left( \text{ref to } query_1 \left\| \text{ref to } qRespAltern_1^? \right) \right\| ... \left\| \left( \text{ref to } query_n \left\| \text{ref to } qRespAltern_n^? \right) \right) \right. \right.$. Upon reception of these objects the *CCP* produces their distribution ready versions, which are $inquiryIO_i^{CCP} = \text{signed}_{K^{-1}_{CCP}}(SigInfo_{CCP} \| inquiryIO_i)$, and $inquiryRespIO_i^{CCP} = \text{signed}_{K^{-1}_{CCP}}(SigInfo_{CCP} \| inquiryRespIO_i)$, respectively.

Examples of an Inquiry IO and of an Inquiry Response IO may be found in section A.3.9 of Annex A.

# 8 Exploitation

## 8.1 Introduction

The P2PTube architecture, described in previous sections, constitutes an efficient, distributed, reliable and secure platform for the interaction between users, and between these and media content or services.

The PLL provides a versatile base platform on which a variety of different, independent, inter-peer interactions may take place, in a secure manner, under the security-wise coordination of the *CCP's* PLL instance. This layer provides the UEL with the necessary services so that it may perform its role, also, in a secure manner and support the safe unfolding of complex distributed operations within its tissue.

This architecture therefore provides a versatile and secure base platform on top of which (by building upon UEL capabilities), multiple different operational patterns may be supported.

The P2PTube architecture may thus be employed by a variety of on-line media distributing entities, lending support to various different types of business models. Specifically, it provides the necessary tools and provisions to support the BMs identified in chapter IV, as the appropriate ones for an on-line media delivering initiative.

The following sub-sections, of section 8, demonstrate how this architecture may effectively be exploited to support said business models and associated content delivery operations.

## 8.2 Advertisement BM Support

When employing an advertisement based BM, a media content delivering initiative obtains its revenue by selling captured user attention, to advertisers. The user's attention is attracted by providing him with some interesting content. That attention is sold to advertisers by, somehow, linking the consumption of said content to the consumption of advertisement messages.

The P2PTube architecture provides optimal support for the "Add supported BM" described in chapter IV, which is especially suited for an on-line medium.
Said BM's most relevant operation (in economic terms), is a user's sale of his attention to an advertiser over the watching of some specific advertisement Media Object (MO). As explained in section 2 of chapter IV, there are two modes to this BM: mandatory unrewarded advertisement viewing; and voluntary rewarded advertisement viewing.

The P2PTube architecture supports both such modes through two different sets of UEL operations which were already explained in section 5.5.8.3.1. In this manner this system enables a robust and secure economic exploitation, of the sale of user attention, to take place.

## 8.3 Donation BM Support

When employing a donation based BM, a media content delivering initiative obtains its revenue through the reception of voluntary donations, from content consuming users. Such initiatives strive to captivate user's good will and translate it into donations.

This type of BM, which is described in chapter IV, is perfectly supported by the P2PTube architecture. Its most relevant operation (in economic terms), is a user's donation, of

monetary, funds to another user and the taxation of that donation, by the content delivering entity.

This monetary funds donation scheme is supported as a set of UEL operations, which may be described in the following manner:

- At an early time, a specific producer user, $U_{prod}$, inserts into the system a specific MO $MO_i^{CCP}$. The MO insertion operation is supported in the simple client/server manner indicated in section 5.5.8.2.1.
- Also at an early time, a specific regular user, $U_i$, loads his system account with some amount of monetary funds. The monetary funds injection operation is supported in the simple client/server manner indicated in section 5.5.8.2.1, where the $CCP$ takes care of all necessary banking operations;
- Some time later, $U_i$ decides to consume $MO_i^{CCP}$. His hosting peer retrieves $MO_i^{CCP}$, from the system in the hybrid manner explained in section 5.5.8.2.2;
- $U_i$ consumes $MO_i^{CCP}$, likes it and decides to reward $U_{prod}$. He specifies $U_{prod}$ and $MO_i^{CCP}$'s identities and the amount to be donated. His hosting peer (e.g. $PP_i$), then proceeds in the client/server manner indicated in section 5.5.8.2.1. That is:
    - $PP_i$ issues a secure request to $CCP$, informing it of the desired donation;
    - $CCP$ checks if $U_i$'s account balance permits the desired donation. If all is ok, the $CCP$ extracts the indicated amount from $U_i$'s account, deposits a pre-determined fraction of it in $U_{prod}$'s account and the rest of it in the operating entity's account. It then sends acknowledgement response back to $PP_i$ for user information.

Through this set of operations the P2PTube architecture thus enables, in a secure fashion, the economic exploitation of inter-user monetary donations, over a hybrid P2P architecture.

## 8.4 Ransom BM Support

The ransom BM is merely a variant of the donation BM. In this variant, donations, from consumer users to producer ones, are performed before the actual MO is published, as a form of financing its production.

The only difference from the set of operations presented in section 8.3, is that user $U_i$ will not initially be able to retrieve and consume $MO_i^{CCP}$. Instead he will only come into contact with a Ransom Announcement for $MO_i^{CCP}$ ($RnsmAnn^{MO_i^{CCP}}$), published by $U_{prod}$, proclaiming the possibility of $MO_i^{CCP}$'s future existence, conditioned to it's *a priori* financing (ransom payment). Furthermore, after a specific cumulative monetary amount (initially specified in $rnsmAnn^{MO_i^{CCP}}$), is achieved (i.e. the ransom is paid), through donations, it no longer is possible to donate to $U_{prod}$, on the grounds of that MO ($MO_i^{CCP}$), and said producer is expected to deliver it, within reasonable time.

The base, UEL, operational scheme supporting this BM is the following:

- At an early time, a specific producer user, $U_{prod}$, inserts into the system a specific Ransom Announcement, $RnsmAnn^{MO_i^{CCP}}$, publicizing the possible future existence of MO $MO_i^{CCP}$, if the system users donate a specific monetary amount ($amount_X$) to $U_{prod}$ on the grounds of $MO_i^{CCP}$. The $RnsmAnn^{MO_i^{CCP}}$ insertion operation is supported in the client/server manner indicated in section 5.5.8.2.1.
- Also at an early time, a specific regular user, $U_i$, securely loads his system account with some amount of monetary funds. The monetary funds injection operation is supported in the client/server manner indicated in section 5.5.8.2.1, where the $CCP$ takes care of all necessary banking operations;
- Some time later, $U_i$ comes across $RnsmAnn^{MO_i^{CCP}}$ (through an MO search operation, supported in the hybrid manner explained in section 5.5.8.2.2), and decides to donate to $U_{prod}$ to finance the production of the $MO_i^{CCP}$. He specifies $U_{prod}$ and $RnsmAnn^{MO_i^{CCP}}$, and identities and the amount to be donated. His hosting peer (e.g. $PP_i$), then proceeds in the client/server manner indicated in section 5.5.8.2.1. That is:
    - $PP_i$ issues a secure request to $CCP$, informing it of the desired donation;
    - $CCP$ checks if $U_i$'s account balance permits the desired donation and if an $ammount_X$ of donations to $U_{prod}$ on the grounds of $MO_i^{CCP}$ ($RnsmAnn^{MO_i^{CCP}}$), as not yet been reached. If all is ok, the $CCP$ extracts the indicated amount from $U_i$'s account, deposits a pre-determined fraction of it in $U_{prod}$'s account and the rest of it in the operating entity's account. It then sends acknowledgement response back to $PP_i$ for user information.

Through this set of operations the P2PTube architecture thus securely enables the economic exploitation of a MO ransoming, over a hybrid P2P architecture.

## 8.5 Traditional BM Support

The traditional BMs supporting the activities of real-world or on-line media content delivering initiatives are typically based on the direct sale of content access. As argued in chapter IV, such BMs are not truly suited for an on-line operation. Nonetheless, the P2PTube architecture is also capable of supporting such business models (if the data model, presented in section 4, is augmented to support the necessary extra event types).

The basic, UEL, operational scheme to support a BM based on the direct sale of content access rights is the following:

- At an early time, a specific producer user, $U_{prod}$, inserts into the system a specific MO ($MO_i^{CCP}$). The MO's media contents are ciphered with a specific secret key, $K_x$. The delivery of the MO and associated key, to the system, is performed in the client/server manner indicated in section 5.5.8.2.1, where the key is included in the sensitive part (see section 5.5.2), of the UEL request message , and the ciphered MO is included in the non-sensitive part of said message;

- Also at an early time, a specific regular user, $U_i$, loads his system account with some amount of monetary funds. The monetary funds injection operation is supported in the simple client/server manner indicated in section 5.5.8.2.1, where the $CCP$ takes care of all necessary banking operations;
- Some time later, $U_i$ decides to consume $MO_i^{CCP}$. His hosting peer, $PP_i$:
  - securely interacts with the $CCP$ (in the simple client/server manner explained in section 5.5.8.2.1), to acquire, for $U_i$, access rights to $MO_i^{CCP}$. If all is ok the $CCP$ handles all the necessary monetary transactions, and sends back to $PP_i$ a data object ($L_{U_i}^{MO_i^{CCP}}$), containing a license granting $U_i$ the right to access $MO_i^{CCP}$ and the necessary key to decrypt the MO's media contents;
  - retrieves $MO_i^{CCP}$, from the system in the hybrid manner explained in section 5.5.8.2.2;
- $PP_i$ inspects the received license and $MO_i^{CCP}$ to verify $U_i$'s right to consume $MO_i^{CCP}$, and if all is ok, (as it presumably should be), it proceeds to decipher $MO_i^{CCP}$'s ciphered parts, and render it for $U_i$'s consumption.

The mentioned license object may be defined as

$$L_{U_i}^{MO_i^{CCP}} = \text{signed}_{K_{CCP}^{-1}}\left(\text{momaniplicense} \| id \| SigInfo_{CCP} \| rightsdef \| u_i \| mo_i^{CCP} \| \left( EncInfo_{U_i} \| enc_{K_{U_i}}(K_x) \right)\right)$$

, where parameter momaniplicense is the identification of the type of the object, $id$ is the license's identifier, $SigInfo_{CCP}$ carries the necessary information to enable the validation of *CCP's* signature of the license, $rightsdef$ is the definition of the specific rights to which the user is entitled, $u_i$ is the identification of the user being given the rights, $mo_i^{CCP}$ is the identifier of the MO over which rights are being granted. The final part of the license is composed by $EncInfo_{U_i}$ (a parameter carrying the necessary information, other than the decryption key, to decode $K_x$), and by the decoding key which is encrypted with the user's public key and may, thus, only be decrypted by the user (with his private key).

Other BMs exist (such as the subscription based one, for instance), which are based on the sale of content access. P2PTube architecture may also enforce such BMs. The main differences between their supporting operational scheme, and the above presented one, are that:

- access purchase (and thus, license emission), will be performed with different periodicity;
- license $L_{U_i}^{MO_i^{CCP}}$ will grant different rights over different sets of MOs;

The P2PTube thus, also, provides also the necessary tools for a secure exploitation of more traditional business models.

## 8.6 Conclusions

As demonstrated in the previous sections, the P2PTube architecture's versatility enables it to securely support the business models identified in chapter IV, while harnessing all the

advantages of delivering content over a hybrid P2P architecture. It is thus equipped with the necessary tools to support successful on-line media delivering initiatives.

# VI Contributions

## 1 Introduction

In this chapter we explain, in a systematic and integrated way, the contributions brought in by the work developed during the course of this thesis and thoroughly described in the previous chapters. In particular we explain how they contribute to the advancement of the current state of the art in the field of safe P2P delivery of rich media content, and in related fields. We also show how the mentioned contributions were subjected to peer analysis and acceptance though scientific publication.

Section 2 explains the value-added achieved by having clearly identified and characterized the on-going techno-economical paradigm change as well as its consequences in terms of the adequacy of new BMs for on-line content distribution, which may be profitable and simultaneously popular among consumers.

Section 3 explains the advantages of the innovative P2PTube architecture, over the current alternatives, in supporting the above-mentioned BMs in a secure and efficient manner. It shows how said architecture combines and expands different pre-existing technologies and concepts to provide capabilities presently missing in the field of P2P content delivery.

Section 4 explains the innovative work conducted towards the development of complex and rich information objects, which are better suited for the new BMs indicated in section 2 and which take better advantage of the potentialities of the P2PTube architecture.

Section 5 lists and contextualizes the publications which resulted from the work that was carried out in the course of this thesis.

Finally, section 6 presents this chapter's concluding remarks.

## 2 Adequate BM Identification

The work exposed in chapter IV builds on the realization, laid out in chapter III, that the characteristics of the on-line medium and of on-line content delivery, are leading to an operational and economic paradigm change in the way media content is distributed.
In chapter IV, we, thus, identify the optimal BMs to support on-line content delivery, within the conditions established by the mentioned emerging paradigm.

The realization and definition that we have performed, of the on-going paradigm change is deeper and clearer than those that have come out of other researches and reflections in this field, and leads us to more radically new conclusions.

In [176] it is realized that the characteristics of information goods production and distribution are different than those of material goods. They clearly understand that the main costs, in this context, are those involved in the production of the first copy of an information good, and that, reproduction costs, (i.e. marginal costs), are approaching zero. Furthermore, the authors of [176], recognize the economic importance of the capture and sale of user attention, especially in a context which is characterized by network effects. They also understand that copyright owners should focus on maximizing the value of their property, and not its access protection.
In the view, of said authors, there are two main commercial strategies that information goods producers may follow:

- striving to become the dominant firm, in a specific product niche, and thus achieve a cost advantage over its competitors, by reducing the average costs of production though increased sales volume;
- striving to maintain a differentiated product from its competitors:

The study in scope, nonetheless, supports scarcity based BMs, were the revenue is predominantly derived though the direct sale of content. Thus, in spite of their recognition of the progressive disappearance of marginal costs, and hence, virtual scarcity, they do not take the next logical step, that we do, in their analysis of reality, and stop short of truly realizing, and appreciating the consequences of, the on-going techno-economical parading shift.

In [177] it is also recognized that online distribution and retail is leading us to a world were media reproduction (copying) and distribution costs are almost non-existent and informational scarcity is annulled. Their conclusion is that the way forward, for content producers/distributors is to “sell less of more”. They argue, that online media stores should exploit the fact that virtual “shelf space” is unlimited, and, thus, greatly extend their offer in order to harness the profits of selling low priced and less demanded media goods, i.e. the “tail goods”. They maintain that such goods are, collectively, profitable enough to sustain distributors’ operations and competitive enough (price-wise), to compete with free distribution (typically pirate distribution).
The authors thus identify the progressive annulment of virtual goods’ scarcity, but still cling onto scarcity dependent BMs, leaning on the use of low pricing schemes to keep the sold goods competitive. This way, the analysis presented in [177], does not really perceive, we believe, to its full extent, the consequences of the on-going paradigm change in the field of information goods’ production and distribution.

In light of the above stated, it becomes evident that the analyses of present reality, in the field in scope, that are exposed in [176] or [177], (as many other similar works), are not as incisive and comprehensive as the one we espouse.
The analysis that we have performed, and exposed in chapter IV, contributes, thus, to a clearer identification and to an expansion of the comprehension of the true dimensions and consequences, of the on-going changes in the area of information goods production and distribution, caused by the digitalization and virtualization of such activities.

The conclusions that we derived have taken us a step beyond the views arguing for the exploitation of the “long tail“, “freemium” BMs, or dynamic pricing schemes, and have lead us to a fuller acceptance of the fact that the main commodity is user attention, fidelity and good will.

Given the deeper, and more radical, realizations that are at their base, the BMs, that we have identified/devised (in chapter IV) for supporting on-line content delivery, are more realistic and better suited to their operational environment (on-line medium), and to the “information access culture” of the Internet, than the predominant BMs in current use in that context.

The BMs, that we have defined, frontally accept the practical elimination of on-line scarcity. They are thus better prepared to deal with that fact than even service-oriented BMs such as subscription based ones.

They also take into greater consideration the importance of user attention, as they perform a more explicit, aggressive, and user satisfactory, acquisition of user attention.

Furthermore, the BM identification and definition work, presented in chapter IV, also contributes with the definition of BMs, which, more clearly, systematically, and unhesitantly

(than current alternatives in the field in scope), bet on donation and ransom based economic gain derivation.

Supporting the above mentioned contributions are the inquiry results analysed in section 3 of chapter IV. Lending further validity, to such contributions, is their recognition by scientific peers by means of their publication, (by this work's authors), in the papers referenced in [166], [178] and [179].

# 3 Necessary P2P Architecture Definition

## 3.1 Introduction

To provide the necessary technical support structure for the BMs described in chapter IV, we defined the P2PTube architecture (exposed in chapter V). It extends and optimizes the capabilities of current P2P technology, to achieve a better overall performance and to endow said technology with dependable reliability and security assuring provisions, which, presently, are typically missing or very poor (in the non-commercial sector), or over complicated, over intrusive and over stretched (in the commercial sector).

The P2PTube architecture thus merges a hybrid P2P operation with PKI-like security provisions and rights management capabilities, to provide a superior support for said BMs.

The added value of this architecture, its contribution to the supporting of the new BMs, resides, more than on any of its individual concepts or solutions, on the devised overall proposal that results from the conjunction of all said concepts and solutions. The P2PTube must thus be comparatively analysed in its entirety, and within the technical and economical context of its operation, so that its contribution becomes more apparent.

Within the above defined context, when compared with the tools and systems exposed in section 2 of chapter II, (predominantly employed in non-commercial scenarios), the P2PTube's advantages consist of a superior efficiency in content delivery and overall system performance, as well as of a greater level of operational security and reliability.

When compared with the technologies exposed in section 3 of chapter II (typically employed in commercial scenarios), the P2PTube's corresponding functionalities present much greater simplicity and are less intrusive on the user's enjoyment of the information goods, while protecting the content owner's contextual rights.

In comparison to the overall technical solutions typically employed by commercial, P2P based, content delivery initiatives, the P2PTube architecture presents advantages in the fields of overall system performance, versatility, seamless integration of security provisions throughout all system sectors, non-intrusiveness, and exploitation of synergies between content delivery and security structures.

Given its, overall, superior security capabilities, better operational performance and non-intrusiveness, the P2PTube architecture is, thus, better equipped and more finely tuned to support the BMs identified in chapter IV, than existing alternatives.

The next sub-sections present each of the above comparisons in a more detailed manner.

## 3.2 Comparison to Current P2P Technology

When compared to the technical solutions approached in chapter II, (typically employed in non-commercial, P2P based, content delivery initiatives), the P2PTube architecture is much more reliable in the discovery and retrieval of content.

Its centralized registry, (at the $CCP$, and at the $CCP$ managed $OCP$ collective), of all the system's Media Objects (MOs), of the semantic information that characterizes each of them, and of the availability for their redistribution by the peripheral peer collective (besides other information), enables:

- semantic based content searches (equivalent to *googling*), to be performed over the entire set of available MOs in a rapid manner without flooding the network with queries and in a single inter-peer interaction (between a $PP$ and the $CCP$ or an $OCP$);
- the optimization of content discovery (locating the media files for retrieval). As the $CCP$, and all $OCPs$, have a global view of the system, they are capable of directing downloading $PPs$ to the most appropriate "server" peers, thus performing an optimized load balancing and eliminating free riding;
- the optimization of content retrieval. For the reasons expressed in the previous bullet, the P2PTube architecture can efficiently distribute the burden of content redistribution over the peer collective thus enabling faster and more reliable media object retrievals;
- the performing of more efficient, reliable and coherent searches over the entire data structure of the system, such as searches over users, peers, etc.

Given the trustworthiness of the $CCP$, and of all $OCPs$, the system's distributed core structure also enables it to provide a number of security facilities which are generally absent from the P2P platforms (especially those employed in the non-commercial sector), such as:

- guaranteeing the confidentiality, authenticity, integrity and non-repudiability of all exchanged messages;
- guaranteeing the authenticity, integrity and non-repudiability of all the MOs, and other information objects distributed across the system (either in a client/server or P2P fashion);
- guaranteeing peer, user and MO identity.

Endowed with the above stated security capabilities, the P2PTube is thus capable of supporting a secure management and transaction of monetary funds.

Even if compared to technical solutions, which employ distributed collective/mutual authentication provisions, as those mentioned in chapter II (e.g. [49] [51]), P2PTube is at an advantage because such solutions (as explained in section 2.4 of chapter II):

- imply a considerable operational overhead – multiple interactions between peers are necessary to achieve some security. In P2PTube this is achieved with far less interactions;
- present considerable weaknesses in the face of sufficiently vast attacks – require that a minimum quorum of "honest" peers is present. No such necessity exists in P2PTube architecture;
- frequently assume the reliability of the network – no such assumption exists in P2PTube architecture;

It may be argued that, as the P2PTube architecture is coordinated by a central entity, it is less scalable than the typical P2P solution (especially in the non-commercial sector). However, this is not so. Fully distributed P2P systems present a vast range of scalability

problems, of their own, which are related to the inter-discovery of peers and to the discovery of content, as the number of peers in the system grows and no entity exists with a global and consistent view of the system's state.
In P2PTube architecture, the $CCP$ eliminates such problems, and, if enough resources are invested into $CCP$, it's centrality will not be a problem, just like the centrality of Wikipedia's and Google's central server farms, is not a problem.

A further advantage of the P2PTube solution resides in its employment of a powerful, versatile and standard tool in all its provisions which deal with serialized information expression. The tool in question is the MPEG-21 standard. It is employed in the expression (structuring) of:

- the PLL and UEL messages exchanged between P2PTube peers;
- all the operational information objects employed and distributed in the system;
- the event report request and corresponding event reports;
- the actual consumable MOs distributed throughout the P2PTube;

MPEG-21's extensive employment renders the P2PTube architecture more standard, than its current counterparts (which frequently do not even follow a clear and public communication standard), easier to expand (given the standard's extensiveness, added capabilities and continuous evolution), and easier to interface with.

For all this, the P2PTube architecture is capable of reliably and securely supporting a number of business models, for the delivery of media content, which the solutions employed in the non-commercial sector of P2P content delivery, clearly cannot, and which the commercial sector solutions are not optimally suited for.

## 3.3 Comparison to Current DRM Capabilities

As explained in section 5.5.9 of chapter V, in the P2PTube architecture, the enforcing of DRM aspects is not handled by isolated components or operations. Instead, its enforcement pervades the system's overall structure and operation.

The P2PTube architecture provides all the necessary security provisions to enable a reliable and trustworthy identification of users, peers and media content, the preservation of MO integrity, the protection of the association between users and the media content that they own, and the collection of information on user actions over content.
It thus safeguards the rights of all users to privacy, to a secure access to their account and user data, and a to a reliable access to MOs. It also protects the rights of content owners as is guards against content corruption, misuse, and wrongful ownership declaration.

In the present context, when compared with the tools and platforms exposed in section 2, of chapter II, the P2PTube architecture naturally presents much higher rights management capabilities, as it possesses much more solid and dependable security provisions (as explained in section 3.2 of the present chapter).

When compared with the technologies exposed in section 3 of chapter II, P2PTube's DRM provisions are lighter and intrude less into the users manipulation of content.
Said provisions have less content access restriction capabilities, but this is really not a shortcoming. P2PTube's DRM-like skills serve different immediate objectives (than those of most technologies presented in section 3 of chapter II) that are dictated by the BMs that the system must support, which do not depend on content access restriction (scarcity creation).

For that, the provisions in scope are not focussed on access restriction but on the assurance of information integrity and of authorship declaration authenticity. Such more realistic

objectives are thus more fully achieved by P2PTube's DRM provisions. These may therefore be simpler, and intrude less into the users' enjoyment of content, without it being a disadvantage. To the contrary, it serves the BMs, defined in chapter IV, better than the solutions presented in section 3 of chapter II, and contributes to fomenting a more cooperative attitude, from users, which may thus be counted upon to actually contribute to the system's DRM activities, by reporting illegitimate content.

## 3.4 Comparison to Current Commercial Platforms

When compared to the typical technical platforms employed in commercial, P2P based, content delivery initiatives (approached in section 4 of chapter II), the P2PTube architecture presents several technical and BM related advantages.

The P2PTube architecture is tailored to support BMs that derive gains in a lateral manner to the actual consumption of media goods, (through advertisement and donations).
In accordance with that, it, therefore, practically does not enforce access control. As already stated in the previous section, this enables the system to maintain a simpler and more efficient operation, and to provide a level of security which is, contextually, better than those of the systems presented in section 4 of chapter II (for instance, the access restrictive, Qtrax or ReelTime).

Also, the P2PTube architecture does not rely on any pre-existing P2P structure (such as Qtrax and iMesh do), which globally is out of the system's control and presents reliability issues. Instead it employs a hybrid structure of its own. Content object location and retrieval are, thus, performed in a P2P fashion, but under the optimizing coordination of the system's, reliable, central provisions ($CCP$ and $OCPs$), which also handle all trust and security demanding tasks and maintain a global and updated view of the system's overall state.
This dedicated structure and its hybridness, therefore, allow the P2PTube architecture to perform such content object location and retrieval tasks, more efficiently than most systems presented in section 4 of chapter II (e.g. Babelgum and Joost). Simultaneously, these characteristics of P2PTube architecture enable it to perform such tasks, in a more reliable way than most such systems (e.g. Qtrax and iMesh).

Furthermore, the full integration between the architecture's security and content delivery provisions enable it to take advantage of synergies between the two. It thus transforms the typical TTP security solutions, employed by commercial initiatives (for security provisioning), into an integral component of the overall architecture, which is able to exploit its P2P capabilities also for security purposes.
For all of the above, the P2PTube architecture is much better equipped, than the technical solutions in scope, to take full advantage of data super-distribution, in a reliable manner.

The P2PTube architecture, when compared to the technical solutions employed by existing commercial, P2P based, content delivery systems, thus presents:

- more efficient and reliable content distributions capabilities;
- more powerful content usage monitoring provisions;
- more flexible content manipulation (by users) permissions;

In light of the above, the P2PTube architecture, thus provides more adequate tools, than the existing, commercially employed, alternatives, to reliably support the opened BMs that we deem necessary (as explained in chapter IV), for a successful operation in the field of commercial P2P content delivery.

## 3.5 Conclusions

Given what was argued in sections 3.2, 3.3 and 3.4 of the present chapter, it may be concluded that the P2PTube architecture offers new and expanded capabilities when compared to existing alternatives in the field of P2P content distribution (both in the commercial and non-commercial sectors).

Its advantages over its non-commercial counterparts stem, basically, from its employment of reliable and trusted central coordinating provisions and of a number of security tools, whose employment is rendered possible by said provisions. An added advantage is the normalization that results from P2PTube's pervasive employment of MPEG-21.

The advantages of P2PTube solution, over its commercial counterparts, stem from its employment of a specific and dedicated hybrid-P2P structure of its own, and the synergistic operation of this structure and of the architecture's security provisions.

Furthermore, the provisions, of the P2PTube architecture, described in section 6 of chapter V, enable the reliable, coherent and secure interoperation between any two, mutually trusted, systems which employ this architecture. This is also an advantage over existing alternatives, especially in the commercial sector, where interoperability problems are commonplace.

For all the above, the P2PTube architecture contributes to the expansion of the state-of-the-art in the field of secure P2P content distribution with opened BMs.

The validation of the P2PTube architecture could, typically, be achieved by means of its implementation and test. That process was, indeed, approached and some of the system's components were effectively developed (such as the cryptographic module for instance). However, the implementation of the entire architecture proved to be an immense task that would inevitably be excessively time consuming. Furthermore, the new mechanisms and architectural solutions, included in P2PTube, are based on some very well established technological building blocks and concepts (hybrid P2P, PKI, asymmetric cryptography, MPEG-21). This lends a basic credibility to our proposals. Therefore, P2PTube's technical implementation is, we believe, not of fundamental importance to demonstrate its overall feasibility, as its basic building blocks are robustly established
In light of the above, the full implementation of the P2PTube architecture was not pursued.

Our validation efforts were, instead, focused on attaining scientific peer recognition of the value of the P2PTube architecture and of the view and concepts that it embodies. In that regard the papers referenced in [180] [181] were published.

# 4 Complex MO Development

## 4.1 Introduction

As it was concluded in chapter IV, online content distribution initiatives, should employ open access BMs to capture user attention, interest and good will. In this context, the distribution of richer media objects will be advantageous as it contributes to such acquisitions.

In accordance with this fact, we also performed innovative work in the development of rich and empowered information objects. This work, even if it is only lateral to the overall thesis exposed in this dissertation, did attain some relevance, and his, for that reason worth mentioning.

The work in scope comprises the definition of the file and (MPEG-21 based) metadata formats of the MOs, distributed though the P2PTube architecture, (presented in chapter V). It also comprises the definition of a (MPEG-21 based) mechanism for the expression of inter MO relationships, which has resulted in an extension to part 3 of the MPEG-21 standard.

## 4.2 MO Format Definition

The defined P2PTube MO format constitutes a novel, complex media object format, with added capabilities when compared to current media file formats, including those presently exchanged in P2P networks.

Said format heavily employs the, open and widely known, MPEG-21 standard. This employment lends the standard's versatility and power to P2PTube's MOs, and facilitates the inclusion of other provisions, in said MOs, that are presented below. The use of MPEG-21 for the structuring of P2P exchanged secure information objects is a novel contribution.

The P2PTube MO format includes the necessary security provisions to enable the validation of the integrity and origin authenticity of such objects. It also includes the tools to enable the precise (RDF based) semantic definition of their content. These features, even if not exclusive to P2PTube's MOs, are attained, in them, in a more powerful and organic way, given the value of the employed standards (MPEG-21 and RDF) and the systemic integration of the MO's structure and the system's overall operation.

The P2PTube MO format thus permits the construction of secure, complex information (media) objects containing multiple different media resources, richly characterized and inter-webbed with RDF based semantic metadata. It is, therefore, a more versatile, standard, self-contained, richer, and more semantically contextualized, object, than those exchanged in current P2P platforms, both in the commercial and non-commercial sector, enabling a richer usage experience.

The above mentioned work, (or its forerunning work), was successfully subjected to peer scrutiny which resulted in the publication of the paper referenced in [182].

## 4.3 Inter MO Relationships Expression Development

In the course of the development of the P2PTube MO format, work was also done on the conception of means to perform the expression of inter-MO relationships. This work was done synergistically with that done within European project CONVERGENCE.

A novel, RDF based, mechanism, was thus defined, to enable the expression of semantic relationships between MPEG-21 information objects. Said mechanism enables a self-contained, clearer and less ambiguous expression of the relational context of MOs. It does so in a more effective and semantically informed manner, than alternate methods (e.g. HTML). Furthermore, it achieves it in alignment with emergent Semantic Web initiatives as the Linked-Data proposal.

The mechanism in scope provides the means for the expression of information which can later be exploited for performing smarter searches over the interrelations between MOs. It also provides the means to trade query precision for recall and vice-versa, and facilitates the decoupling of content from location by replacing static links between object locations (the typical http links), with semantically rich relationships between information objects. It, thus, semantically and relationally, empowers MPEG-21 enabling the development of a complex, yet precise and easily interpretable, overall MO fabric.

The, above described, work was validated by peer recognition by means of its publication in the papers referenced in [183] and [192]. It also originated the submission of a proposal, to the MPEG-21 consortium, for the extension of part 3 of the MPEG-21 standard, with the relational capabilities in scope. Said proposal has been accepted and is now part of an international standard.
The specific MPEG-21 standardization contribution documents are those referenced in [184], [185], [186] and [187].

# 5 Publications

In the context of the development of this PhD work, several publications were achieved, focusing on different aspects of said work. Those publications have already been briefly addressed in the previous sections of this chapter. It is, however, relevant to present them in a more detailed and contextualized manner. Below we include a table which lists the publications in question, in a contextualized manner.

Table 5 – Publications

| | | |
|---|---|---|
| Conference Paper | Paper Reference | **Helder Castro**, A. Pimenta Alves, "*Support for Media Content Production and Distribution in the Internet Era*", First Workshop on Interdisciplinary Research in New Media, 2007. |
| | Focused Aspect of Phd Work | This paper presents a long term analysis of the socio-economical evolution of the support structures for information content production. The realities that it reveals and the conclusions at which it arrives, helped guide and fuel the reflexion and analysis present in chapters III and IV. |
| | | |
| Conference Paper | Paper Reference | **H. Castro**, M. T. Andrade, A. P. Alves, "*Governed Media Distribution based on Nonrestrictive DRM*", International Conference on Telecommunications and Multimedia, Ierapetra, Crete, Greece, 2008. |
| | Focused Aspect of Phd Work | This paper presents work, developed synergistically with European project (ENTHRONE), which reflects the analysis presented in chapters III and IV and which provided some basic ideas for the for the P2PTube architecture, exposed in chapter V. |
| | | |
| Conference Paper | Paper Reference | **H. Castro**, A. P. Alves, "*Cognitive Object Format*", International Conference on Knowledge Engineering and Ontology Development, Funchal, Madeira, Portugal, 2009. |
| | Focused Aspect of Phd Work | This paper presents work pertaining to the development of complex and versatile information objects, to increase the attractiveness of all-digital, on-line content consumption. This work's contributions are addressed in section 4 of chapter VI. |
| | | |
| Journal Paper | Paper Reference | **Helder Castro**, Artur P. Alves, Carlos Serrão, Brett Caraway, "*A New Paradigm for Content Producers*", IEEE MultiMedia, vol. 17, no. 2, pp. 90-93, Apr. 2010. |
| | Focused Aspect of Phd Work | This paper presents the reflexion and analysis that we have performed, and the conclusions that we have derived, regarding the on-going changes, in the field of information distribution, with the development of on-line delivery. Its content is present in chapters III and IV. |
| | | |

| | | |
|---|---|---|
| Conference Paper | Paper Reference | **Helder Castro**, Maria Teresa Andrade, Fernando Almeida, Giuseppe Tropea, Nicola Blefari Melazzi,Leonardo Chiariglione, Aziz S. Mousas and Dimitra I. Kaklaman, "*Exploring Semantic Relationships Across Internet Resources*", NWeSP 2011, Spain, 2011. |
| | Focused Aspect of Phd Work | This paper presents work pertaining to the development of complex and versatile information objects, to increase the attractiveness of all-digital, on-line content manipulation. It's development was synergistic with the work done in the European project CONVERGENCE. The developed work's contributions are addressed in section 4 of chapter VI. |
| | | |
| Conference Paper | Paper Reference | **Helder Castro**, Artur P. Alves, "*A P2P Content Delivery System for Alternative Business Models*", NWeSP 2011, Spain, 2011. |
| | Focused Aspect of Phd Work | This paper presents some of the analysis exposed in chapters III and IV. It then builds on said analysis in order to define a P2P content delivery system which is adequate for the BMs defined in chapter IV, and whose basic architectural options are reflected in chapter V. |
| | | |
| Journal Paper | Paper Reference | **H. Castro**, M.T. Andrade, F. Almeida, G. Troppea, N.B. Melazzi, A.S. Mousas, and D.I. Kaklamani, "*Semantically Connected Web Resources with MPEG-21*", Springer Multimedia Tools and Applications (submitted). |
| | Focused Aspect of Phd Work | This paper presents work pertaining to the development of complex and versatile information objects, to increase the attractiveness of all-digital, on-line content manipulation. It focuses on the definition and exploitation of a mechanism conceived to enable the explicit expression of inter-Digital Item relationships in MPEG-21. Said mechanism was accepted as an extension to the MPEG-21 DII and is now part of that international standard. It's development was synergistic with the work done in the European project CONVERGENCE. This work's contributions are addressed in section 4.3 of chapter VI. |
| | | |
| Journal Paper | Paper Reference | **H. Castro**, A. P. Alves and M. T. Andrade, "*Reliable P2P Content Delivery for Alternative Business Models*", International Journal of Computer Information Systems and Industrial Management Applications, vol. 5, pp. 11–29, 2013 (accepted). |
| | Focused Aspect of Phd Work | This paper presents the key aspects of the P2P architecture that we have conceived to securely support the BMs, which, we argue, are adequate to sustain the on-line distribution of information content. Its contents are included in chapter V. |

# 6 Conclusions

As the previous sections of the present chapter show, the overall contribution of this thesis, to the advancement of safe content delivery over P2P networks, is not a single, individualized, point or proposal. It is instead composed by a set of interrelated and interdependent parts, as depicted in Figure 64.

At the root of the contribution is the realization, that was performed, of the true depth of the occurring changes (brought on by on-line, all-digital content delivery), in the world of

information content production and distribution. This realization shaped the posterior course of our work, as it revealed that safety, in a content distribution operation, is, first and foremost, dependent of the economical/business aspects of that operation. That is the reason why said realization, and the consequent identification of adequate BMs to operate in the on-line medium, play the root role, in our contributions.

Building on the earlier contribution component, is a second such component which consists of the definition of a P2P architecture containing the necessary/missing security and coordination provisions to adequately support the enforcement of the above mentioned BMs.

A third and final component, whose need and role is identified by the previous two, consists of the conception/development of a complex information object format, which enables a more powerful manipulation of such objects by the system, and a more pleasant usage, of them, by the users.

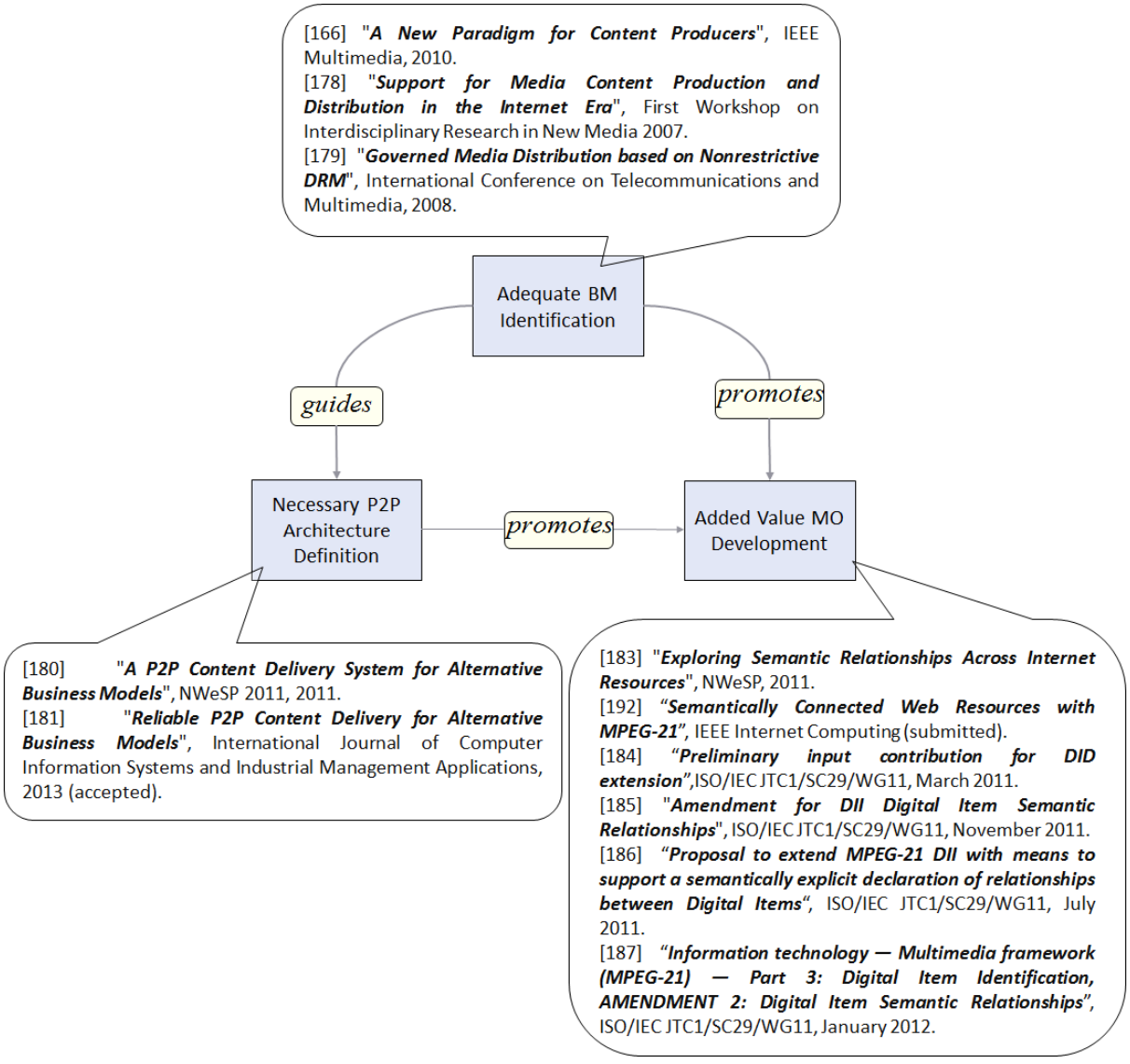


Figure 64 – Contribution Interrelations and Publications

Figure 64 also indicates (references) the papers which publish the work done on each of the contribution components.

# VII Final Remarks

Throughout the previous chapters we have presented an interrelated set of reflections, analyses and proposals, whose full value can best be appreciated only when they are observed collectively. This means that the defence of that value is not a simple matter as it requires validating each research component and their logical interconnection.

This complex and broad reaching nature, that our work acquired, was not the fruit of choice but was, instead, demanded by the characteristics of the problems that we set ourselves to address. Our search for a safe P2P distribution of media content has thus leads us through various fields of research, which, we believe, we have managed to successfully traverse to reach valid and innovative solutions.

In spite of all the work that was developed there is still ample room for further advancements. In our opinion the most relevant venues for the continuation and concretization of this work are the following:

- Advancing the architectural solution, presented in chapter V:
    - so that the unique *CCP*, may be replaced by a group of *CCPs*, which divide the central core workload amongst themselves, and enable a more concurrent “writing” access to the system’s data structure, while preserving its coherence;
    - in order to enable a finer differentiation of the roles of core peers, so that finer grained responsibility and workload distributions can be achieved;
    - so as to further empower the system’s periphery so that it becomes able to distribute, amongst itself, progressively broader parts of the system’s data structure, whiteout breaching system security and that structure’s coherence;
    - in order for it to accommodate the temporary operation of the system (more accurately, the system’s periphery), in the advent of the (temporary) complete failure of the CCP or of the entire core;
- Further enrichment of the MO format to enable a more fine grained binding of semantic concepts to specific “locations” of the media resources;
- Continuation of the studies of user opinion on different on-line content access modes and BMs in order to further fine tune the BMs identified in chapter IV;
- Implementation and testing of the P2PTube architecture.

We hope that, in this dissertation, we have managed to communicate a clear idea of the problems that we addressed, of the solutions that we proposed and of the interdependencies between said problems and solutions.

# Annex A – Data Objects

## A.1 IPCL Data Objects

### A.1.1 *IPCLMHFile*

Figure 65 presents an example of an IPCL Message Head File.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:pllm="urn:p2pt:pllm"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="ipclm">
    <Descriptor id="ipclm_srcPIP">
      <Statement mimeType="text/plain">
        IP address of the source peer
      </Statement>
    </Descriptor>
    <Descriptor id="ipclm_dstPIP">
      <Statement mimeType="text/plain">
        IP address of the destination peer
      </Statement>
    </Descriptor>
    <Item id="ipclm_content">
      <Component>
        <Resource mimeType="application/tar" ref="PLLMFile name"/>
      </Component>
    </Item>
  </Container>
</DIDL>
```

Figure 65 – IPCLMHFile Example

## A.2 PLL Data Objects

### A.2.1 *PLL Message*

#### A.2.1.1 *PLLSIHFile*

Figure 66 presents an example of a PLL Sensitive Info Head File.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:pllm="urn:p2pt:pllm"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="pllm_pllsih">
    <Descriptor id="encinfodescr">
      <Statement mimeType="text/xml">
        <DIDL id="encinfo">
          <Container>
            <Item id="pid">
              <Component>
                <Resource mimeType="text/plain">
                  identification of the encoding peer
```

```
                    </Resource>
                  </Component>
                </Item>
                <Item id="algdef">
                  <Component>
                    <Resource mimeType="text/plain">
                      definition of the encoding algorithm
                    </Resource>
                  </Component>
                </Item>
                <Item id="pubk">
                  <Descriptor id="pubkid">
                    <Statement mimeType="text/plain">
                      identifier of the public key
                    </Statement>
                  </Descriptor>
                  <Component>
                    <Resource mimeType="text/plain">
                      public key
                    </Resource>
                  </Component>
                </Item>
                <Item id="privk">
                  <Descriptor id="privk">
                    <Statement mimeType="text/plain">
                      identifier of the private key
                    </Statement>
                  </Descriptor>
                </Item>
                <Item id="commSessID">
                  <Component >
                    <Resource mimeType="text/plain">
                      id of the communication session
                    </Resource>
                  </Component>
                </Item>
              </Container>
            </DIDL>
          </Statement>
        </Descriptor>
        <Item id="pllm_securecontent">
          <Component>
            <Resource mimeType="application/tar" ref="EncodedFileName"/>
          </Component>
        </Item>
      </Container>
</DIDL>
```

Figure 66 – PLLSIHFile Example

### A.2.1.2 *PLLPSIHFile*

Figure 67 presents an example of a PLL Protected Sensitive Info Head File.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:pllm="urn:p2pt:pllm"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="pllm_pllpsih">
```

```
      <Descriptor id="signature">
        <Statement mimeType="text/xml">
          <dsig:Signature>
            ..........
          </dsig:Signature>
        </Statement>
      </Descriptor>
      <Item id="pllm_pllpsi_signed_content">
        <Item id="pllm_pllpsic">
          <Component>
            <Resource mimeType="text/xml" ref="PLLPSICFileName"/>
          </Component>
        </Item>
      </Item>
    </Container>
</DIDL>
```

Figure 67 – PLLPSIHFile Example

### **A.2.1.3** *PLLPSICTopFile*

Figure 68 presents an example of a PLL Protected Sensitive Info Core Top File.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:pllm="urn:p2pt:pllm"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="pllm_pllsic">
    <Item id="siginfo">
      <Component>
        <Resource>
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
                    public key
                  </Resource>
                </Component>
              </Item>
```

```
                <Item id="privk">
                  <Descriptor id="privk">
                    <Statement mimeType="text/plain">
                      identifier of the private key
                    </Statement>
                  </Descriptor>
                </Item>
              </Container>
            </DIDL>
          </Resource>
        </Component>
      </Item>
      <Item id="serial">
        <Item id="previousSerialPart">
          <Item id="sessionID">
            <Component>
              <Resource mimeType="text/plain">
                session id
              </Resource>
            </Component>
          </Item>
          <Item id="tstamp">
            <Component>
              <Resource mimeType="text/plain">
                YYYY-MM-DDThh:mmTZD
              </Resource>
            </Component>
          </Item>
          <Item id="msgcount">
            <Item id="type">
              <Component>
                <Resource mimeType="text/plain">
                  req
                </Resource>
              </Component>
            </Item>
            <Item id="count">
              <Component>
                <Resource mimeType="text/plain">
                  value of the message counter
                </Resource>
              </Component>
            </Item>
          </Item>
        </Item>
        <Item id="newSerialPart">
          <Item id="sessionID">
            <Component>
              <Resource mimeType="text/plain">
                session id
              </Resource>
            </Component>
          </Item>
          <Item id="tstamp">
            <Component>
              <Resource mimeType="text/plain">
                YYYY-MM-DDThh:mmTZD
              </Resource>
            </Component>
          </Item>
```

```
            <Item id="msgcount">
              <Item id="type">
                <Component>
                  <Resource mimeType="text/plain">
                    resp
                  </Resource>
                </Component>
              </Item>
              <Item id="count">
                <Component>
                  <Resource mimeType="text/plain">
                    value of the message counter
                  </Resource>
                </Component>
              </Item>
            </Item>
          </Item>
        </Item>
      </Container>
</DIDL>
```

Figure 68 – PLLPSICTopFile Example

### A.2.2 *Peer Registration Certificate*

Figure 69 presents an example of a peripheral peer's registration certificate.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:prc="urn:p2pt:prc"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="prc">
    <Descriptor id="prc_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="prc_content">
      <Descriptor id="id">
        <Statement mimeType="text/xml">
          <dii:Identifier>
            p2ptube:io:rcertif001
          </dii:Identifier>
        </Statement>
      </Descriptor>
      <Descriptor id="prc_validTerm">
        <Statement mimeType="text/plain">
          "YYYY-MM-DDThh:mmTZD"
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
```

```
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
                    public key
                  </Resource>
                </Component>
              </Item>
              <Item id="privk">
                <Descriptor id="privk">
                  <Statement mimeType="text/plain">
                    identifier of the private key
                  </Statement>
                </Descriptor>
              </Item>
            </Container>
          </DIDL>
        </Statement>
      </Descriptor>
      <Item id="sid">
        <Descriptor id="peerIDInfo_signature">
          <Statement mimeType="text/xml">
            <dsig:Signature>
              ....
            </dsig:Signature>
          </Statement>
        </Descriptor>
        <Item id="peerIDInfo">
          <Item id="pid">
            <Component>
              <Resource mimeType="text/plain">
                identification of the peer
              </Resource>
            </Component>
          </Item>
          <Item id="algdef">
            <Component>
              <Resource mimeType="text/plain">
                definition of the peer's signing algorithm
              </Resource>
            </Component>
          </Item>
          <Item id="pubk">
            <Descriptor id="pubkid">
              <Statement mimeType="text/plain">
```

```
                    identifier of the peer's public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
                    public key
                  </Resource>
                </Component>
              </Item>
              <Item id="privk">
                <Descriptor id="privk">
                  <Statement mimeType="text/plain">
                    identifier of the peer's private key
                  </Statement>
                </Descriptor>
              </Item>
            </Item>
          </Item>
          <Item id="prc_peerRole">
            <Component>
              <Resource mimeType="text/plain">
                peripheral
              </Resource>
            </Component>
          </Item>
        </Item>
      </Container>
</DIDL>
```

Figure 69 – Peer Registration Certificate Example

### A.2.3 *PLL Peer Info Object*

Figure 70 presents an example of a PLL Peer Info Object.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:pio="urn:p2pt:pio"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="pio">
    <Descriptor id="pio_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          ....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="pio_content">
      <Descriptor id="id">
        <Statement mimeType="text/xml">
          <dii:Identifier>
            p2ptube:io:pio001
          </dii:Identifier>
        </Statement>
      </Descriptor>
      <Descriptor id="vTerm">
        <Statement mimeType="text/plain">
          YYYY-MM-DDThh:mmTZD
```

```
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
                    public key
                  </Resource>
                </Component>
              </Item>
              <Item id="privk">
                <Descriptor id="privk">
                  <Statement mimeType="text/plain">
                    identifier of the private key
                  </Statement>
                </Descriptor>
              </Item>
            </Container>
          </DIDL>
        </Statement>
      </Descriptor>
      <Item id="peerIP">
        <Component>
          <Resource mimeType="text/plain">
            .....
          </Resource>
        </Component>
      </Item>
      <Item id="peerRegCertif">
        <Component>
          <Resource mimeType="text/xml">
            <DIDL>
              .....
            </DIDL>
          </Resource>
        </Component>
      </Item>
    </Item>
```

```
  </Container>
</DIDL>
```

Figure 70 – Peer Info Object Example

### A.2.4 *Peer Quarantine List*

Figure 71 presents an example of a Peer Quarantine List.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:pql="urn:p2pt:pql"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="pql">
    <Descriptor id="pql_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          ....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="pql_content">
      <Descriptor id="vTerm">
        <Statement mimeType="text/plain">
          YYYY-MM-DDThh:mmTZD
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
                    public key
                  </Resource>
                </Component>
              </Item>
              <Item id="privk">
```

```
                    <Descriptor id="privk">
                      <Statement mimeType="text/plain">
                        identifier of the private key
                      </Statement>
                    </Descriptor>
                  </Item>
                </Container>
              </DIDL>
            </Statement>
          </Descriptor>
          <Item id="quarantinedPeer1">
            <Component>
              <Resource mimeType="text/plain">
                peer 1 id
              </Resource>
            </Component>
          </Item>
          .............
          <Item id="quarantinedPeerN">
            <Component>
              <Resource mimeType="text/plain">
                peer N id
              </Resource>
            </Component>
          </Item>
        </Item>
      </Container>
</DIDL>
```

Figure 71 – Peer Quarantine List Example

### A.2.5 *PLL Info Retrieval Permit*

Figure 72 presents an example of a PLL Info Retrieval Permit.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:plliorp="urn:p2pt: plliorp"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="pirp">
    <Descriptor id="pirp_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          ....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="pirp_content">
      <Descriptor id="id">
        <Statement mimeType="text/xml">
          <dii:Identifier>
            p2ptube:io:pirp001
          </dii:Identifier>
        </Statement>
      </Descriptor>
      <Descriptor id="objectType">
        <Statement mimeType="text/plain">
          ppInfoRetrPermit
```

```
    </Statement>
  </Descriptor>
  <Descriptor id="vTerm">
    <Statement mimeType="text/plain">
      YYYY-MM-DDThh:mmTZD
    </Statement>
  </Descriptor>
  <Descriptor id="siginfodescr">
    <Statement mimeType="text/xml">
      <DIDL id="siginfo">
        <Container>
          <Item id="pid">
            <Component>
              <Resource mimeType="text/plain">
                identification of the signing peer
              </Resource>
            </Component>
          </Item>
          <Item id="algdef">
            <Component>
              <Resource mimeType="text/plain">
                definition of the signing algorithm
              </Resource>
            </Component>
          </Item>
          <Item id="pubk">
            <Descriptor id="pubkid">
              <Statement mimeType="text/plain">
                identifier of the public key
              </Statement>
            </Descriptor>
            <Component>
              <Resource mimeType="text/plain">
                public key
              </Resource>
            </Component>
          </Item>
          <Item id="privk">
            <Descriptor id="privk">
              <Statement mimeType="text/plain">
                identifier of the private key
              </Statement>
            </Descriptor>
          </Item>
        </Container>
      </DIDL>
    </Statement>
  </Descriptor>
  <Item id="queryingPeerInfo">
    <Item id="queryingPeerID">
      <Component>
        <Resource mimeType="text/plain">
          p2ptube:ppeer:xpto001
        </Resource>
      </Component>
    </Item>
  </Item>
  <Item id="searchedInfoDef">
    .....
  </Item>
```

```
        <Item id="pllildoID">
          <Component>
            <Resource mimeType="text/plain">
              p2ptube:ppeer:pllildo001
            </Resource>
          </Component>
        </Item>
      </Item>
    </Container>
</DIDL>
```

Figure 72 – PLL Info Object Retrieval Permit Example

# A.3 UEL Data Objects

## A.3.1 *UEL Message*

### A.3.1.1 *UELSIFile*

Figure 73 presents an example of a UEL Sensitive Info File.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:uelsif="urn:p2pt:uelsif"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="uelm_uelsi">
    <Descriptor id="usignature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="uelsif_content">
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="uid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing user
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
```

```
                        <Resource mimeType="text/plain">
                            public key
                        </Resource>
                    </Component>
                </Item>
                <Item id="privk">
                    <Descriptor id="privk">
                        <Statement mimeType="text/plain">
                            identifier of the private key
                        </Statement>
                    </Descriptor>
                </Item>
            </Container>
        </DIDL>
    </Statement>
</Descriptor>
<Descriptor id="serial">
    <Statement mimeType="text/xml">
        .....
    </Statement>
</Descriptor>
<Item id="uels_info">
    <Item id="uelmsgtype">
        <Component>
            <Resource mimeType="text/xml">
                .....
            </Resource>
        </Component>
    </Item>
    <Item id="further_uels_info_param1">
        <Component>
            <Resource mimeType="text/xml">
                .....
            </Resource>
        </Component>
    </Item>
    .....
    <Item id="further_uels_info_paramn">
        <Component>
            <Resource mimeType="text/xml">
                .....
            </Resource>
        </Component>
    </Item>
</Item>
</Item>
</Container>
</DIDL>
```

Figure 73 – UELSIFile Example

### A.3.2 *User Registration Certificate*

Figure 74 presents an example of a User Registration Certificate.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:urc="urn:p2pt:urc"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
```

```
        didl.xsd">
  <Container id="urc">
    <Descriptor id="urc_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="urc_content">
      <Descriptor id="id">
        <Statement mimeType="text/xml">
          <dii:Identifier>
            p2ptube:io:rcertifu001
          </dii:Identifier>
        </Statement>
      </Descriptor>
      <Descriptor id="vTerm">
        <Statement mimeType="text/plain">
          YYYY-MM-DDThh:mmTZD
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
                    public key
                  </Resource>
                </Component>
              </Item>
              <Item id="privk">
                <Descriptor id="privk">
                  <Statement mimeType="text/plain">
                    identifier of the private key
                  </Statement>
                </Descriptor>
              </Item>
            </Container>
```

```
            </DIDL>
          </Statement>
        </Descriptor>
        <Item id="sidU_item">
          <Component>
            <Resource mimeType="text/xml">
              <DIDL id="sidU">
                <Container>
                  <Descriptor id="uInfoSignature">
                    <Statement mimeType="text/xml">
                      <dsig:Signature>
                        .....
                      </dsig:Signature>
                    </Statement>
                  </Descriptor>
                  <Item id="userInfo">
                    <Item id="uid">
                      <Component>
                        <Resource mimeType="text/plain">
                          identification of the user
                        </Resource>
                      </Component>
                    </Item>
                    <Item id="uname">
                      <Component>
                        <Resource mimeType="text/plain">
                          username
                        </Resource>
                      </Component>
                    </Item>
                    <Item id="algdef">
                      <Component>
                        <Resource mimeType="text/plain">
                          definition of the user's signing algorithm
                        </Resource>
                      </Component>
                    </Item>
                    <Item id="pubk">
                      <Descriptor id="pubkid">
                        <Statement mimeType="text/plain">
                          identifier of the user's public key
                        </Statement>
                      </Descriptor>
                      <Component>
                        <Resource mimeType="text/plain">
                          public key
                        </Resource>
                      </Component>
                    </Item>
                    <Item id="privk">
                      <Descriptor id="privk">
                        <Statement mimeType="text/plain">
                          identifier of the user's private key
                        </Statement>
                      </Descriptor>
                    </Item>
                  </Item>
                </Container>
              </DIDL>
            </Resource>
```

```
            </Component>
          </Item>
        </Item>
    </Container>
</DIDL>
```

Figure 74 – User Registration Certificate Example

### A.3.3 *User Hosting Certificate*

Figure 75 presents an example of a User Hosting Certificate.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
        xmlns:upc="urn:p2pt:upc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
        xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
        didl.xsd">
  <Container id="uhc">
    <Descriptor id="uhc_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="uhc_content">
      <Descriptor id="id">
        <Statement mimeType="text/xml">
          <dii:Identifier>
            p2ptube:io:uhostcertif001
          </dii:Identifier>
        </Statement>
      </Descriptor>
      <Descriptor id="vTerm">
        <Statement mimeType="text/plain">
          "YYYY-MM-DDThh:mmTZD"
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
```

```
                        identifier of the public key
                    </Statement>
                </Descriptor>
                <Component>
                    <Resource mimeType="text/plain">
                        public key
                    </Resource>
                </Component>
            </Item>
            <Item id="privk">
                <Descriptor id="privk">
                    <Statement mimeType="text/plain">
                        identifier of the private key
                    </Statement>
                </Descriptor>
            </Item>
        </Container>
    </DIDL>
</Statement>
</Descriptor>
<Item id="hostPeerInfo">
    <Item id="hostPeerID">
        <Component>
            <Resource mimeType="text/plain">
                p2ptube:ppeer:xpto001
            </Resource>
        </Component>
    </Item>
</Item>
<Item id="rcertifU">
    .....
</Item>
</Item>
</Container>
</DIDL>
```

Figure 75 – User Hosting Certificate Example

### A.3.4 *Search Query Response Object*

Figure 76 presents an example of a Search Query Response Object.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:sqro="urn:p2pt:sqro"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      didl.xsd">
  <Container id="sqro">
    <Descriptor id="sqro_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="sqro_content">
      <Descriptor id="id">
        <Statement mimeType="text/xml">
          <dii:Identifier>
            p2ptube:io:sqro001
```

```
            </dii:Identifier>
        </Statement>
    </Descriptor>
    <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
            <DIDL id="siginfo">
                <Container>
                    <Item id="pid">
                        <Component>
                            <Resource mimeType="text/plain">
                                identification of the signing peer
                            </Resource>
                        </Component>
                    </Item>
                    <Item id="algdef">
                        <Component>
                            <Resource mimeType="text/plain">
                                definition of the signing algorithm
                            </Resource>
                        </Component>
                    </Item>
                    <Item id="pubk">
                        <Descriptor id="pubkid">
                            <Statement mimeType="text/plain">
                                identifier of the public key
                            </Statement>
                        </Descriptor>
                        <Component>
                            <Resource mimeType="text/plain">
                                public key
                            </Resource>
                        </Component>
                    </Item>
                    <Item id="privk">
                        <Descriptor id="privk">
                            <Statement mimeType="text/plain">
                                identifier of the private key
                            </Statement>
                        </Descriptor>
                    </Item>
                </Container>
            </DIDL>
        </Statement>
    </Descriptor>
    <Descriptor id="sqro_emissiontime">
        <Statement mimeType="text/plain">
            YYYY-MM-DDThh:mmTZD
        </Statement>
    </Descriptor>
    <Descriptor id="sqro_answeredquery">
        <Statement mimeType="text/xml">
            <DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
            xmlns:sqro="urn:p2pt:sqro"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
            xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
            didl.xsd">
                <Container id="answeredQ">
                    <Descriptor id="answeredQ_signature">
                        <Statement mimeType="text/xml">
```

```
            <dsig:Signature>
              .....................
            </dsig:Signature>
          </Statement>
        </Descriptor>
        <Item>
          <Descriptor id="id">
            <Statement mimeType="text/xml">
              <dii:Identifier>
                p2ptube:io:answeredQ001
              </dii:Identifier>
            </Statement>
          </Descriptor>
          <Descriptor id="siginfodescr">
            <Statement mimeType="text/xml">
              <DIDL id="siginfo">
                <Container>
                  <Item id="pid">
                    <Component>
                      <Resource mimeType="text/plain">
                        identification of the signing peer
                      </Resource>
                    </Component>
                  </Item>
                  <Item id="algdef">
                    <Component>
                      <Resource mimeType="text/plain">
                        definition of the signing algorithm
                      </Resource>
                    </Component>
                  </Item>
                  <Item id="pubk">
                    <Descriptor id="pubkid">
                      <Statement mimeType="text/plain">
                        identifier of the public key
                      </Statement>
                    </Descriptor>
                    <Component>
                      <Resource mimeType="text/plain">
                        public key
                      </Resource>
                    </Component>
                  </Item>
                  <Item id="privk">
                    <Descriptor id="privk">
                      <Statement mimeType="text/plain">
                        identifier of the private key
                      </Statement>
                    </Descriptor>
                  </Item>
                </Container>
              </DIDL>
            </Statement>
          </Descriptor>
          <Component>
            <Resource mimeType="text/plain">
              query string
            </Resource>
          </Component>
        </Item>
```

```
                </Container>
              </DIDL>
            </Statement>
          </Descriptor>
          <Component id="sqro_molist">
            <Resource mimeType="text/xml">
              <sqro:MOList>
                .....................
              </sqro:MOList>
            </Resource>
          </Component>
        </Item>
      </Container>
</DIDL>
```

Figure 76 – SQRO Example

### A.3.5 *UEL Information Location Describing Object*

Figure 77 presents an example of a UELILDO Object.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
        xmlns:uelildo="urn:p2pt:uelildo"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
        xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
        didl.xsd">
  <Container id="uelildo">
    <Descriptor id="uelildo_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>.....</dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="uelildo_content">
      <Descriptor id="id">
        <Statement mimeType="text/xml">
          <dii:Identifier>
            p2ptube:io:uelildo001
          </dii:Identifier>
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
```

```
                    <Descriptor id="pubkid">
                      <Statement mimeType="text/plain">
                        identifier of the public key
                      </Statement>
                    </Descriptor>
                    <Component>
                      <Resource mimeType="text/plain">
                        public key
                      </Resource>
                    </Component>
                  </Item>
                  <Item id="privk">
                    <Descriptor id="privk">
                      <Statement mimeType="text/plain">
                        identifier of the private key
                      </Statement>
                    </Descriptor>
                  </Item>
                </Container>
              </DIDL>
            </Statement>
          </Descriptor>
          <Descriptor id="uelildo_emissiontime">
            <Statement mimeType="text/plain">
              YYYY-MM-DDThh:mmTZD
            </Statement>
          </Descriptor>
          <Item id="uelildo_loclist">
            <Item id="hostPeer1Info">
              <Item id="hostPeerID">
                <Component>
                  <Resource mimeType="text/plain">
                    p2ptube:ppeer:xpto001
                  </Resource>
                </Component>
              </Item>
            </Item>
            .....
            <Item id="hostPeerNInfo">
              <Item id="hostPeerID">
                <Component>
                  <Resource mimeType="text/plain">
                    p2ptube:ppeer:xpto00n
                  </Resource>
                </Component>
              </Item>
            </Item>
          </Item>
          <Item id="uelildo_fraglist">
            <Component>
              <Resource mimeType="text/xml">
                <DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                  xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS didl.xsd">
                  <Container id="fraglist">
                    <Descriptor id="fraglist_signature">
                      <Statement mimeType="text/plain">
                        YYYY-MM-DDThh:mmTZD
                      </Statement>
```

```
                </Descriptor>
                <Item id="fraglist_content">
                  <Descriptor id="fraglist_id">
                    <Statement mimeType="text/xml">
                      <dii:Identifier>
                        p2ptube:io:fraglist001
                      </dii:Identifier>
                    </Statement>
                  </Descriptor>
                  <Descriptor id="siginfodescr">
                    <Statement mimeType="text/xml">
                      <DIDL id="siginfo">
                        <Container>
                          <Item id="pid">
                            <Component>
                              <Resource mimeType="text/plain">
                                identification of the signing peer
                              </Resource>
                            </Component>
                          </Item>
                          <Item id="algdef">
                            <Component>
                              <Resource mimeType="text/plain">
                                definition of the signing algorithm
                              </Resource>
                            </Component>
                          </Item>
                          <Item id="pubk">
                            <Descriptor id="pubkid">
                              <Statement mimeType="text/plain">
                                identifier of the public key
                              </Statement>
                            </Descriptor>
                            <Component>
                              <Resource mimeType="text/plain">
                                public key
                              </Resource>
                            </Component>
                          </Item>
                          <Item id="privk">
                            <Descriptor id="privk">
                              <Statement mimeType="text/plain">
                                identifier of the private key
                              </Statement>
                            </Descriptor>
                          </Item>
                        </Container>
                      </DIDL>
                    </Statement>
                  </Descriptor>
                  <Item id="moID">
                    <Component>
                      <Resource mimeType="text/plain">
                        p2ptube:mo:abc1
                      </Resource>
                    </Component>
                  </Item>
                  <Item id="fragDefs">
                    <Item id="fragDef1">
                      <Descriptor id="fragID">
```

```
      <Statement mimeType="text/xml">
        p2ptube:mo:abc1:frag0
      </Statement>
    </Descriptor>
    <Descriptor id="fragSize">
      <Statement mimeType="text/xml">
        size of the fragment
      </Statement>
    </Descriptor>
    <Descriptor id="siginfodescr">
      <Statement mimeType="text/xml">
        <DIDL id="siginfo">
          <Container>
            <Item id="pid">
              <Component>
                <Resource mimeType="text/plain">
                  identification of the signing peer
                </Resource>
              </Component>
            </Item>
            <Item id="algdef">
              <Component>
                <Resource mimeType="text/plain">
                  definition of the signing algorithm
                </Resource>
              </Component>
            </Item>
            <Item id="pubk">
              <Descriptor id="pubkid">
                <Statement mimeType="text/plain">
                  identifier of the public key
                </Statement>
              </Descriptor>
              <Component>
                <Resource mimeType="text/plain">
                  public key
                </Resource>
              </Component>
            </Item>
            <Item id="privk">
              <Descriptor id="privk">
                <Statement mimeType="text/plain">
                  identifier of the private key
                </Statement>
              </Descriptor>
            </Item>
          </Container>
        </DIDL>
      </Statement>
    </Descriptor>
    <Descriptor id="fragSignature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Component>
      <Resource mimeType="application/p2ptmo" ref="p2ptube:mo:abc1"/>
      <Anchor>
```

```
                                <Descriptor>
                                    <Statement mimeType="text/plain">
                                        p2ptube:mo:abc1:frag0
                                    </Statement>
                                </Descriptor>
                                <Fragment fragmentId="offset(0,100000)"/>
                            </Anchor>
                        </Component>
                    </Item>
                    .....
                    <Item id="fragDefN">
                        <Descriptor id="fragID">
                            <Statement mimeType="text/xml">
                                p2ptube:mo:abc1:fragN
                            </Statement>
                        </Descriptor>
                        <Descriptor id="fragSize">
                            <Statement mimeType="text/xml">
                                size of the fragment
                            </Statement>
                        </Descriptor>
                        <Descriptor id="fragSignature">
                            <Statement mimeType="text/xml">
                                <dsig:Signature>
                                    ..........
                                </dsig:Signature>
                            </Statement>
                        </Descriptor>
                        <Component>
                            <Resource mimeType="application/p2ptmo" ref="p2ptube:mo:abc1"/>
                            <Anchor>
                                <Descriptor>
                                    <Statement mimeType="text/plain">
                                        p2ptube:mo:abc1:fragN
                                    </Statement>
                                </Descriptor>
                                <Fragment fragmentId="offset(100000,200000)"/>
                            </Anchor>
                        </Component>
                    </Item>
                </Item>
            </Item>
        </Container>
    </DIDL>
</Resource>
</Component>
</Item>
</Item>
</Container>
</DIDL>
```

Figure 77 – UELILDO Example

### A.3.6 *UEL Info Retrieval Permit*

Figure 78 presents an example of an UEL Info Retrieval Permit.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
```

```
        xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
        didl.xsd">
  <Container id="uelirp">
    <Descriptor id="uelirp_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="uelirp_content">
      <Descriptor id="uelirp_permtype_declaration">
        <Statement mimeType="text/plain">
          UEL Info Retrieval Permit
        </Statement>
      </Descriptor>
      <Descriptor id="uelirp_tstamp">
        <Statement mimeType="text/plain">
          YYYY-MM-DDThh:mmTZD
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
          .....
      </Descriptor>
      <Item id="enabledPeerInfo">
        <Item id="enabledPeerID">
          <Component>
            <Resource mimeType="text/plain">
              p2ptube:ppeer:xpto001
            </Resource>
          </Component>
        </Item>
      </Item>
      <Item id="infodef">
        <Item id="moID">
          <Component>
            <Resource mimeType="text/plain">
              p2ptube:mo:abc1
            </Resource>
          </Component>
        </Item>
      </Item>
      <Item id="uelildo">
        <Component>
          <Resource mimeType="text/xml">
            <DID>
              .....
            </DID>
          </Resource>
        </Component>
      </Item>
    </Item>
  </Container>
</DIDL>
```

Figure 78 – UEL Info Retrieval Permit Example

### A.3.7 *Media Objects*

Figure 79 presents an example of a *MOTHFile*'s contents.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
        xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
        didl.xsd">
  <Container id="moth">
    <Descriptor id="moth_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="moth_content">
      <Descriptor id="moid">
        <Statement mimeType="text/xml">
          <dii:Identifier>
            p2ptube:io:mo001
          </dii:Identifier>
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
                      public key
                  </Resource>
                </Component>
              </Item>
              <Item id="privk">
                <Descriptor id="privk">
                  <Statement mimeType="text/plain">
                    identifier of the private key
                  </Statement>
                </Descriptor>
              </Item>
            </Container>
          </DIDL>
        </Statement>
      </Descriptor>
      <Descriptor id="tstamp">
```

```
            <Statement mimeType="text/plain">
                YYYY-MM-DDThh:mmTZD
            </Statement>
        </Descriptor>
        <Descriptor id="moi_signature">
            <Statement mimeType="text/xml">
                <dsig:Signature>
                    .....
                </dsig:Signature>
            </Statement>
        </Descriptor>
        <Component id="moiRef">
            <Resource mimeType="application/tar" ref="MOIFileName"/>
        </Component>
    </Item>
  </Container>
</DIDL>
```

Figure 79 – MOTHFile Example

Figure 80 presents an example of a *MOIHFile*'s contents.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
        xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
        didl.xsd">
  <Container id="moih">
    <Descriptor id="moih_u_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="moih_content">
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="uid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing user
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
```

```
                    <Resource mimeType="text/plain">
                      public key
                    </Resource>
                  </Component>
                </Item>
                <Item id="privk">
                  <Descriptor id="privk">
                    <Statement mimeType="text/plain">
                      identifier of the private key
                    </Statement>
                  </Descriptor>
                </Item>
              </Container>
            </DIDL>
          </Statement>
        </Descriptor>
        <Descriptor id="ownerUserID">
          <Statement mimeType="text/plain">
            p2ptube:user:abc1
          </Statement>
        </Descriptor>
        <Descriptor id="moih_semantics">
          <Statement mimeType="text/xml">
            <rdf:RDF>
              .....
            </rdf:RDF>
          </Statement>
        </Descriptor>
        <Descriptor id="moih_rights">
          <Statement mimeType="text/xml">
            <r:license>
              .....
            </r:license>
          </Statement>
        </Descriptor>
        <Item id="moicFilesInfo">
          <Item id="moic1">
            <Descriptor id="moic1_u_signature">
              <Statement mimeType="text/xml">
                <dsig:Signature>
                  .....
                </dsig:Signature>
              </Statement>
            </Descriptor>
            <Component>
              <Resource mimeType="video/mpeg" ref="MOIC1_FileName"/>
            </Component>
          </Item>
          ……. further Items
          <Item id="moicn">
            <Descriptor id="moicn_u_signature">
              <Statement mimeType="text/xml">
                <dsig:Signature>
                  .....
                </dsig:Signature>
              </Statement>
            </Descriptor>
            <Component>
              <Resource mimeType="video/mpeg" ref="MOICN_FileName"/>
            </Component>
```

```
            </Item>
          </Item>
        </Item>
      </Container>
</DIDL>
```

Figure 80 – MOIHFile Example

### A.3.8 *ERRs and ERs*

Figure 81 presents an example of a P2PTube DID carrying an ERR (the actual ERR content is adapted from [102]).

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
        xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
        xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
        xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS">
        xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS didl.xsd">
  <Container id="err">
    <Descriptor id="err_cp_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          .....................
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="err_content">
      <Descriptor id="err_ID">
        <Statement mimeType="text/plain">
          <dii:Identifier>p2ptube:err:0001</dii:Identifier>
        </Statement>
      </Descriptor>
      <Descriptor id="siginfodescr">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
```

```
                    public key
                  </Resource>
                </Component>
              </Item>
              <Item id="privk">
                <Descriptor id="privk">
                  <Statement mimeType="text/plain">
                    identifier of the private key
                  </Statement>
                </Descriptor>
              </Item>
            </Container>
          </DIDL>
        </Statement>
      </Descriptor>
      <Item id="err_proper">
        <Component>
          <Resource mimeType="text/xml">
            <erl:ERR xmlns="urn:mpeg:mpeg21:2005:01-ERL-NS"
              xmlns:sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
              xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
              xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
              <ERRDescriptor>
                <LifeTime>
                  <StartTime>2011-07-01T00:00:00</StartTime>
                  <EndTime>2011-07-08T00:00:00</EndTime>
                </LifeTime>
                <Modification>
                  <PeerId>.....</PeerId>
                  <UserId>.....</UserId>
                  <Time>2011-06-30T18:15:00</Time>
                  <Description>Creation of ER-R by Bob@acme.org</Description>
                </Modification>
                <Priority>1</Priority>
              </ERRDescriptor>
              <ERSpecification>
                <Identifier xmlns="urn:mpeg:mpeg21:2002:01-DII-NS">urn:mpegRA:mpeg21:dii:eid:1702</Identifier>
                <ERDescription>This is a description of the ER</ERDescription>
                <AccessControl/>
                <ERPayloadSpecification>
                  <ERIdentifier baseId="true">p2ptube:err:0001:er</ERIdentifier>
                  <PeerId/>
                  <UserId/>
                  <Time/>
                  <Location/>
                  <DIOperation/>
                  <DomainData reportTag="Name" semantics="acme:PhoneName" syntax="xsd:String"/>
                  <DomainData semantics="AcmeFilm:length" syntax="xsd:Integer" value="155"/>
                  <DIMetadata>
                    <DISelection>
                      <DISelectionViaDII>
                        urn:mpegra:mpeg21:dii:isrc:BE-R45-98-03948576
                      </DISelectionViaDII>
                    </DISelection>
                    <DIMetadataElement tagName="Title"/>
                    <DIMetadataElement tagName="Artist"/>
                    <DIMetadataElement tagName="ISWC"/>
                  </DIMetadata>
                </ERPayloadSpecification>
                <ERFormatSpecification>
```

```
            <Ref>http://www.acme.org/schemas/phones.xsd</Ref>
          </ERFormatSpecification>
          <ERDeliverySpecification>
            <Recipient>
              <PeerId>p2ptube:ppeer:ccp</PeerId>
              <UserId>.....</UserId>
            </Recipient>
            <DeliveryTime>
              <SpecificTime>
                <AfterOn>2012-07-06T00:00:00</AfterOn>
                <BeforeOn>2012-09-06T00:00:00</BeforeOn>
              </SpecificTime>
            </DeliveryTime>
            <DITransportService>
              <r:serviceReference>
                <sx:wsdlComplete>
                  <sx:wsdl>
                    <nonSecureIndirect URI="http://www.acme.org/ER-wsdlfile.xml"/>
                  </sx:wsdl>
                  <sx:service>er:SendERService</sx:service>
                  <sx:portType>er:SendERPortType</sx:portType>
                </sx:wsdlComplete>
              </r:serviceReference>
            </DITransportService>
          </ERDeliverySpecification>
          <EmbeddedERR>
            <ERRReference>mpeg:mpeg21:dii:ERRID:010</ERRReference>
          </EmbeddedERR>
        </ERSpecification>
        <EventConditionDescriptor>
          <TimeCondition>
            <TimeEvent>
              <SpecificTime>
                <AfterOn>2012-01-01T00:00:00</AfterOn>
                <BeforeOn>2012-01-31T00:00:00</BeforeOn>
              </SpecificTime>
            </TimeEvent>
            <Operator Name="OR"/>
            <TimeEvent>
              <PeriodicTime>
                <Start>2012-07-06T00:00:00</Start>
                <Period>P2M10D</Period>
                <Duration>P1D</Duration>
                <End>2013-07-06T00:00:00</End>
              </PeriodicTime>
            </TimeEvent>
          </TimeCondition>
          <Operator Name="AND"/>
          <DIOperationCondition>
            <DIOperationEvent>
              <Operation>REL:Play</Operation>
              <DII>mpeg:mpeg21:dii:ID:ACME-010</DII>
            </DIOperationEvent>
          </DIOperationCondition>
          <Operator Name="AND"/>
          <PeerCondition>
            <PeerEvent Name="=" Location="infix">
              <mpeg7:Region>au</mpeg7:Region>
            </PeerEvent>
            <Operator Name="OR"/>
```

```
                        <PeerEvent Name="=" Location="infix">
                            <mpeg7:Region>kr</mpeg7:Region>
                        </PeerEvent>
                    </PeerCondition>
                </EventConditionDescriptor>
            </erl:ERR>
        </Resource>
    </Component>
</Item>
<Item id="peerInfo">
    <Item id="peerID">
        <Component>
            <Resource mimeType="text/plain">
                p2ptube:ppeer:xpto001
            </Resource>
        </Component>
    </Item>
</Item>
</Container>
</DIDL>
```

Figure 81 – ERR DID Example

Figure 82 presents an example of a P2PTube DID carrying an ER (the actual ER content is adapted from [102]).

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
      xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
      xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS">
      xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS didl.xsd">
  <Container id="er">
    <Descriptor id="er_pp_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>
          ......................
        </dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="er_content">
      <Descriptor id="siginfodescrB">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
```

```
                </Item>
                <Item id="pubk">
                  <Descriptor id="pubkid">
                    <Statement mimeType="text/plain">
                      identifier of the public key
                    </Statement>
                  </Descriptor>
                  <Component>
                    <Resource mimeType="text/plain">
                      public key
                    </Resource>
                  </Component>
                </Item>
                <Item id="privk">
                  <Descriptor id="privk">
                    <Statement mimeType="text/plain">
                      identifier of the private key
                    </Statement>
                  </Descriptor>
                </Item>
              </Container>
            </DIDL>
          </Statement>
        </Descriptor>
        <Descriptor id="er_u_signature">
          <Statement mimeType="text/xml">
            <dsig:Signature>
              .....
            </dsig:Signature>
          </Statement>
        </Descriptor>
        <Item id="er">
          <Descriptor id="id">
            <Statement mimeType="text/xml">
              <dii:Identifier>
                p2ptube:io:er001
              </dii:Identifier>
            </Statement>
          </Descriptor>
          <Descriptor id="siginfodescrC">
            <Statement mimeType="text/xml">
              <DIDL id="siginfo">
                <Container>
                  <Item id="pid">
                    <Component>
                      <Resource mimeType="text/plain">
                        identification of the signing user
                      </Resource>
                    </Component>
                  </Item>
                  <Item id="algdef">
                    <Component>
                      <Resource mimeType="text/plain">
                        definition of the signing algorithm
                      </Resource>
                    </Component>
                  </Item>
                  <Item id="pubk">
                    <Descriptor id="pubkid">
                      <Statement mimeType="text/plain">
```

```
                        identifier of the public key
                    </Statement>
                </Descriptor>
                <Component>
                    <Resource mimeType="text/plain">
                        public key
                    </Resource>
                </Component>
            </Item>
            <Item id="privk">
                <Descriptor id="privk">
                    <Statement mimeType="text/plain">
                        identifier of the private key
                    </Statement>
                </Descriptor>
            </Item>
        </Container>
    </DIDL>
  </Statement>
</Descriptor>
<Item id="err_ID">
  <Component>
    <Resource mimeType="text/plain">
      p2ptube:err:0001
    </Resource>
  </Component>
</Item>
<Item>
  <Component>
    <Resource mimeType="text/xml">
      <erl:ER xmlns="urn:mpeg:mpeg21:2005:01-ERL-NS"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
        <ERDescriptor>
          <Description>
            A Free-text description of the ER
          </Description>
          <Recipient>
            <PeerId>p2ptube:ppeer:xpto001</PeerId>
            <UserId>.....</UserId>
          </Recipient>
          <Status value="true"/>
          <Modification>
            <PeerId>.....</PeerId>
            <UserId>.....</UserId>
            <Time>2005-11-03T01:22:30</Time>
          </Modification>
          <ERSource>
            <ERRReference>mpeg:mpeg21:dii:ERRID:342</ERRReference>
          </ERSource>
        </ERDescriptor>
        <ERData>
          <PeerId>.....</PeerId>
          <UserId>.....</UserId>
          <Time>2005-11-02T05:11:32</Time>
          <Location>
            <mpeg7:Region>au</mpeg7:Region>
          </Location>
          <DIOperation>REL:Play</DIOperation>
          <ReportedDomainData semantics="acme:PhoneName">
```

```
                        <Name>AcmeModel231-ANS</Name>
                      </ReportedDomainData>
                      <ReportedDomainData>
                        .....
                      </ReportedDomainData>
                      <ReportedDIMetadata>
                        <Title>A very catchy Tune</Title>
                        <Artist>The hippest guys on the Block</Artist>
                        <ISWC>T-345246800-1</ISWC>
                      </ReportedDIMetadata>
                    </ERData>
                  </erl:ER>
                </Resource>
              </Component>
            </Item>
          </Item>
        </Item>
      </Container>
</DIDL>
```

Figure 82 – ER DID Example

### A.3.9 *Inquiry and Inquiry Response IOs*

Figure 83 presents an example of an Inquiry IO.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
      xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS">
  <Container id="inqio_ccp">
    <Descriptor id="inqio_ccp_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>.....</dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="inqio_ccp_content">
      <Descriptor id="siginfodescrCCP">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
```

```
                    <Descriptor id="pubkid">
                      <Statement mimeType="text/plain">
                        identifier of the public key
                      </Statement>
                    </Descriptor>
                    <Component>
                      <Resource mimeType="text/plain">
                        public key
                      </Resource>
                    </Component>
                  </Item>
                  <Item id="privk">
                    <Descriptor id="privk">
                      <Statement mimeType="text/plain">
                        identifier of the private key
                      </Statement>
                    </Descriptor>
                  </Item>
                </Container>
              </DIDL>
            </Statement>
          </Descriptor>
          <Component>
            <Resource mimeType="text/xml">
              <DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
              xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
              xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
              xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS">
                <Container id="inqio">
                  <Descriptor id="inqio_signature">
                    <Statement mimeType="text/xml">
                      <dsig:Signature>.....</dsig:Signature>
                    </Statement>
                  </Descriptor>
                  <Item id="inqio_content">
                    <Descriptor id="id">
                      <Statement mimeType="text/xml">
                        <dii:Identifier>
                          p2ptube:io:inqio001
                        </dii:Identifier>
                      </Statement>
                    </Descriptor>
                    <Descriptor id="siginfodescrU">
                      <Statement mimeType="text/xml">
                        <DIDL id="siginfo">
                          <Container>
                            <Item id="pid">
                              <Component>
                                <Resource mimeType="text/plain">
                                  identification of the signing user
                                </Resource>
                              </Component>
                            </Item>
                            <Item id="algdef">
                              <Component>
                                <Resource mimeType="text/plain">
                                  definition of the signing algorithm
```

```
                    </Resource>
                  </Component>
                </Item>
                <Item id="pubk">
                  <Descriptor id="pubkid">
                    <Statement mimeType="text/plain">
                      identifier of the public key
                    </Statement>
                  </Descriptor>
                  <Component>
                    <Resource mimeType="text/plain">
                      public key
                    </Resource>
                  </Component>
                </Item>
                <Item id="privk">
                  <Descriptor id="privk">
                    <Statement mimeType="text/plain">
                      identifier of the private key
                    </Statement>
                  </Descriptor>
                </Item>
              </Container>
            </DIDL>
          </Statement>
        </Descriptor>
        <Item id="p2ptube:io:inqio001:queryBlock1">
          <Item id="p2ptube:io:inqio001:queryBlock1:query">
            <Component>
              <Resource mimeType="text/plain">
                ..... query ....
              </Resource>
            </Component>
          </Item>
          <Item id="p2ptube:io:inqio001:queryBlock1:responseAlterns">
            <Item id="p2ptube:io:inqio001:queryBlock1:responseAlterns:respAltern1">
              <Component>
                <Resource mimeType="text/plain">
                  ..... resp altern 1 ....
                </Resource>
              </Component>
            </Item>
            .....
            <Item id="p2ptube:io:inqio001:queryBlock1:responseAlterns:respAlternN">
              <Component>
                <Resource mimeType="text/plain">
                  ..... resp altern N ....
                </Resource>
              </Component>
            </Item>
          </Item>
        </Item>
        .......
        <Item id="p2ptube:io:inqio001:queryBlockN">
          <Item id="p2ptube:io:inqio001:queryBlockN:query">
            <Component>
              <Resource mimeType="text/plain">
                ..... query ....
              </Resource>
            </Component>
```

```
                    </Item>
                    <Item id="p2ptube:io:inqio001:queryBlockN:responseAlternatives">
                      <Item id="p2ptube:io:inqio001:queryBlockN:responseAlterns:respAltern1">
                        <Component>
                          <Resource mimeType="text/plain">
                            ..... resp altern 1 ....
                          </Resource>
                        </Component>
                      </Item>
                      .....
                      <Item id="p2ptube:io:inqio001:queryBlockN:responseAlterns:respAlternN">
                        <Component>
                          <Resource mimeType="text/plain">
                            ..... resp altern N ....
                          </Resource>
                        </Component>
                      </Item>
                    </Item>
                  </Item>
                </Container>
              </DIDL>
            </Resource>
          </Component>
        </Item>
      </Container>
</DIDL>
```

Figure 83 – Inquiry IO Example

Figure 84 presents an example of an Inquiry Response IO.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
      xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
      xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS">
  <Container id="inqrio_ccp">
    <Descriptor id="inqrio_ccp_signature">
      <Statement mimeType="text/xml">
        <dsig:Signature>.....</dsig:Signature>
      </Statement>
    </Descriptor>
    <Item id="inqrio_ccp_content">
      <Descriptor id="siginfodescrCCP">
        <Statement mimeType="text/xml">
          <DIDL id="siginfo">
            <Container>
              <Item id="pid">
                <Component>
                  <Resource mimeType="text/plain">
                    identification of the signing peer
                  </Resource>
                </Component>
              </Item>
              <Item id="algdef">
                <Component>
                  <Resource mimeType="text/plain">
```

```
                    definition of the signing algorithm
                  </Resource>
                </Component>
              </Item>
              <Item id="pubk">
                <Descriptor id="pubkid">
                  <Statement mimeType="text/plain">
                    identifier of the public key
                  </Statement>
                </Descriptor>
                <Component>
                  <Resource mimeType="text/plain">
                    public key
                  </Resource>
                </Component>
              </Item>
              <Item id="privk">
                <Descriptor id="privk">
                  <Statement mimeType="text/plain">
                    identifier of the private key
                  </Statement>
                </Descriptor>
              </Item>
            </Container>
          </DIDL>
        </Statement>
      </Descriptor>
      <Component>
        <Resource mimeType="text/xml">
          <DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
          xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
          xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
          xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS">
            <Container id="inqrio">
              <Descriptor id="inqrio_signature">
                <Statement mimeType="text/xml">
                  <dsig:Signature>.....</dsig:Signature>
                </Statement>
              </Descriptor>
              <Item id="inqrio_content">
                <Descriptor id="id">
                  <Statement mimeType="text/xml">
                    <dii:Identifier>
                      p2ptube:io:inqrio001
                    </dii:Identifier>
                  </Statement>
                </Descriptor>
                <Descriptor id="siginfodescrU">
                  <Statement mimeType="text/xml">
                    <DIDL id="siginfo">
                      <Container>
                        <Item id="pid">
                          <Component>
                            <Resource mimeType="text/plain">
                              identification of the signing user
                            </Resource>
                          </Component>
```

```
                    </Item>
                    <Item id="algdef">
                      <Component>
                        <Resource mimeType="text/plain">
                          definition of the signing algorithm
                        </Resource>
                      </Component>
                    </Item>
                    <Item id="pubk">
                      <Descriptor id="pubkid">
                        <Statement mimeType="text/plain">
                          identifier of the public key
                        </Statement>
                      </Descriptor>
                      <Component>
                        <Resource mimeType="text/plain">
                          public key
                        </Resource>
                      </Component>
                    </Item>
                    <Item id="privk">
                      <Descriptor id="privk">
                        <Statement mimeType="text/plain">
                          identifier of the private key
                        </Statement>
                      </Descriptor>
                    </Item>
                  </Container>
                </DIDL>
              </Statement>
            </Descriptor>
            <Item id="p2ptube:io:inqrio001:resp1">
              <Descriptor id="p2ptube:io:inqrio001:resp1:answeredQueryID">
                <Statement mimeType="text/xml">
                  <dii:Identifier>
                    p2ptube:io:inqio001:queryBlock1:query
                  </dii:Identifier>
                </Statement>
              </Descriptor>
              <Item id="p2ptube:io:inqrio001:resp1:correctRespAlternID">
                <Component>
                  <Resource mimeType="text/plain">
                    p2ptube:io:inqio001:queryBlock1:responseAlterns:respAltern?
                  </Resource>
                </Component>
              </Item>
            </Item>
            .....
            <Item id="p2ptube:io:inqrio001:respN">
              <Descriptor id="p2ptube:io:inqrio001:respN:answeredQueryID">
                <Statement mimeType="text/xml">
                  <dii:Identifier>
                    p2ptube:io:inqio001:queryBlockN:query
                  </dii:Identifier>
                </Statement>
              </Descriptor>
              <Item id="p2ptube:io:inqrio001:respN:correctRespAlternID">
                <Component>
                  <Resource mimeType="text/plain">
                    p2ptube:io:inqio001:queryBlockN:responseAlterns:respAltern?
```

```
                                </Resource>
                            </Component>
                        </Item>
                    </Item>
                </Item>
            </Container>
        </DIDL>
    </Resource>
</Component>
</Item>
</Container>
</DIDL>
```

Figure 84 – Inquiry Response IO Example

# Annex B – P2P Technologies

## B.1 Structured P2P Lookup Protocols

### B.1.1 *Content Addressable Network*

CAN [188] is a distributed infrastructure that provides hash table-like functionality, on an Internet-like scale, for the mapping of file names to their location in the network. It does so by supporting the insertion, lookup, and deletion of (key, value) pairs in said table.
The key space (and corresponding values), is divided into zones and individual CAN nodes store a part (a zone), of the hash table data (corresponding to a key sub-space), and also some information about the adjacent zones.

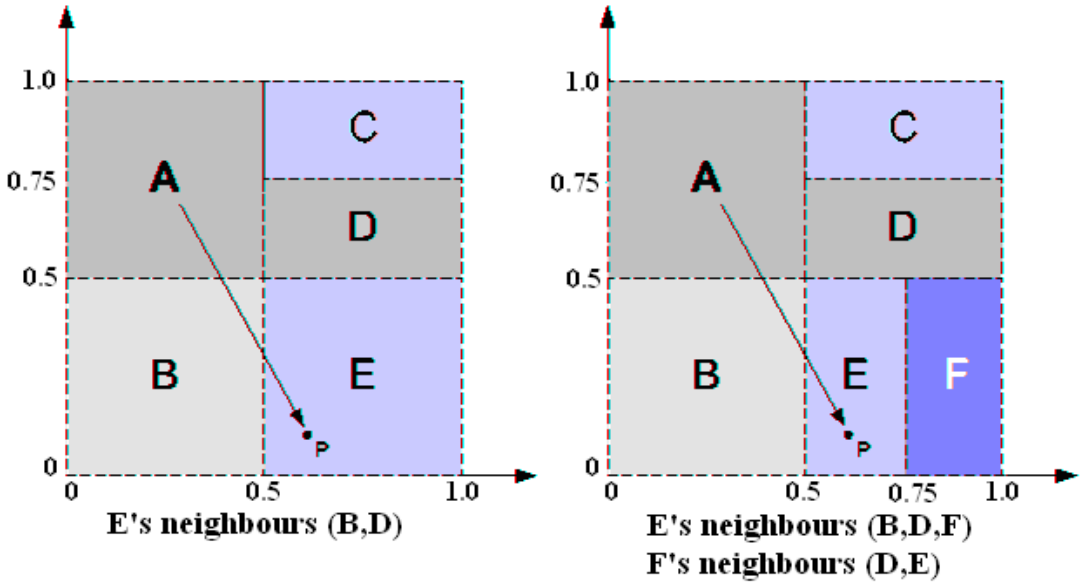


Figure 85 – CAN Coordinate Space divided between 5 zones (left side image) and 6 zones (right side image) (adapted from [2])

CAN defines a virtual d-dimensional Cartesian coordinate space. Each hash table zone corresponds to a section of the coordinate space. Every key *K* in the key space is deterministically mapped onto a point *P* in the coordinate space. A (*K*, *V*) pair is then stored at the node which is responsible for handling the zone that point *P* belongs to (for instance in Figure 85, a (*K*,*V*) pair whose key is mapped to coordinate (0.1,0.2) would be stored in zone B, that is, in the peer handling that zone).

CAN nodes may retrieve a (*K, V*) pair simply by applying the same deterministic function to *K* in order to obtain *P* and then retrieve the corresponding value *V* from the node handling the zone containing P.

If *P* is not located in the requesting node's zone, a request for the *V* stored at *P* must be routed through the system, from node to node until it reaches the node covering *P*.
For the handling of such routing procedures, CAN nodes possess a routing table containing the IP addresses for the nodes handling the adjacent zones to their own.

In the CAN system, the routing of requests is performed by following a straight line path through the Cartesian space from source to destination coordinates (for instance in Figure 85, a request from node A for a key mapped to point *P* would be routed though nodes A, B, E, through the straight line represented by the arrow). Within this structure, requests to

insert, lookup or delete a particular (*K, V*) pair are routed via adjacent zones to the node handling the zone containing the *P* to which the involved key maps.

When a new node joins the CAN system, it is attributed a portion of the coordinate space. That zone is defined by dividing in half the space allocated to an existing node (right side image of Figure 85). This procedure begins with the selection of an already existing node by the newcomer node (through a bootstrapped mechanism). Then, employing the CAN routing mechanism, the adhering node randomly selects a point *P* in the coordinate space and issues a *JOIN* request to the node handling that point. The zone handled by the latter node is then split, and half of it is attributed to the new node.
The new node then proceeds to construct its routing table with the IP addresses of its neighbours. The nodes adjacent to the split zone are also informed of the division so that they update their routing tables to include the new node.

Under ideal conditions, when a node leaves the CAN system, its handled zone and the associated hash table entries are explicitly handed over to one of its neighbouring nodes. Still, under normal conditions, what happens is that nodes send periodic update messages to all of their neighbouring nodes declaring their zone coordinates, their list of neighbours and the coordinates of their neighbour's zones. When prolonged absences of such update messages occur, the neighbouring nodes assume that a failure has happened, and undertake a procedure for the taking over of the abandoned zone.
In the case where several neighbouring nodes fail simultaneously, an expanding ring search mechanism is commenced by one of the operating neighbouring nodes, in order to identify other functioning nodes outside the abandoned region.

The overall design of the CAN system may nonetheless be improved. The main envisioned improvements are:

- Use of multi-dimensional coordinate space – If the number of dimensions of the CAN coordinate space is increased, the routing path length and latency is diminished;

- Multiple coordinate spaces – Each node in the system is assigned a different zone in each coordinate space. Each coordinate space is a "reality". Therefore, for a CAN system with r realities, a single node has r coordinate zones attributed to it, one on every reality and has r independent sets of neighbours. If the contents of the hash table are replicated on every reality, data availability and routing fault tolerance are greatly improved since that in the case of a routing or node failure on one reality, messages can continue to be routed and contents reached using the remaining realities;

- Employment of more advanced CAN routing metrics – The request routing metric for the CAN system may be improved by taking into account the underlying IP topology by having peers measure the network-level round-trip-time to each of its neighbours. A message is then forwarded, for a specific destination, to the neighbour with the maximum ratio of progress to round-trip-time. This procedure would diminish path latency;

- Overloading coordinate zones – This consists of permitting multiple nodes to share the same zone. In such a case a node maintains a list of its peers as well as a neighbour list. This offers such advantages as: reduced path length and latency, because increasing the number of nodes per zone results in the reduction of the number of nodes in the system; reduced per-hop latency; improved fault tolerance because a zone is only abandoned when all the nodes in it, leave simultaneously;

- Multiple hash functions – If k different hash functions are used to map a single key onto k points in the coordinate space and thus replicate the corresponding (key, value) pair at k distinct nodes in the system, the content's availability will be increased. Furthermore queries for a specific hash table entry may also be simultaneously sent to all k nodes thus reducing the average query latency;

- Topologically sensitive construction of the CAN overlay network – This consists of taking into account the underlying IP network topology for the allocation of nodes to zones;

- Uniform Coordinate Space Partitioning – According to CAN, when a new node joins, some randomly selected existing node will split its zone in two, and one of the portions will be attributed to the new node. This may be improved if instead the existing occupant node first compares the volume of its zone with those of its immediate neighbours, and then the zone with the largest volume is split to accommodate the new node. This more uniform partitioning of the coordinate space permits a better load balancing;

- Caching and Replication techniques for "hot spot" management – Popular (key, value) pairs in a CAN system may be made more widely available by employing some caching and replication techniques commonly applied to the Web:

    - Caching – Maintenance of a cache of the data keys most recently accessed by a CAN node. Before forwarding a request for a data key towards its destination, a node first checks its cache. This way the number of caches from which a data key may be retrieved from increases in direct proportion to its popularity and thus popular content becomes more widely available;

    - Replication – A CAN node overloaded with requests for a particular data key replicates it at each of its neighbouring nodes.

### B.1.2 *Chord*

Chord [189] is a distributed lookup protocol for the efficient location of data objects in P2P structures. This protocol employs deterministically generated keys as file and node identifiers, and it performs the mapping of file keys into node keys in order to determine the location for the storage of the (key, file) pair, which is the mapped node.

A node identifier is derived by hashing the node's IP address. A key identifier is calculated by hashing the data key.

The node identifying keys are ordered in a modulo $2^m$ "identifier circle". File key k is attributed to the first node whose identifier is equal to or follows k in the identifier space. This is termed the successor node of key k.
Figure 86, for instance, presents an identifier circle with m=3. The successor of the identifier 6 is peer 0, so key 6 will be located at the node with an id value of 0.

Within the Chord scheme, the only routing information that nodes need to possess is the location of its successor node on the circle. Queries are forwarded around the circle via these successor pointers up to the point where a node which possesses the key is encountered. This is the node the query maps to.

Upon adhesion of a new node *n* to the network, certain keys previously assigned to *n*'s successor will be assigned to *n*. Upon *n*'s removal from the network, all of its keys attributed are reassigned to its successor.

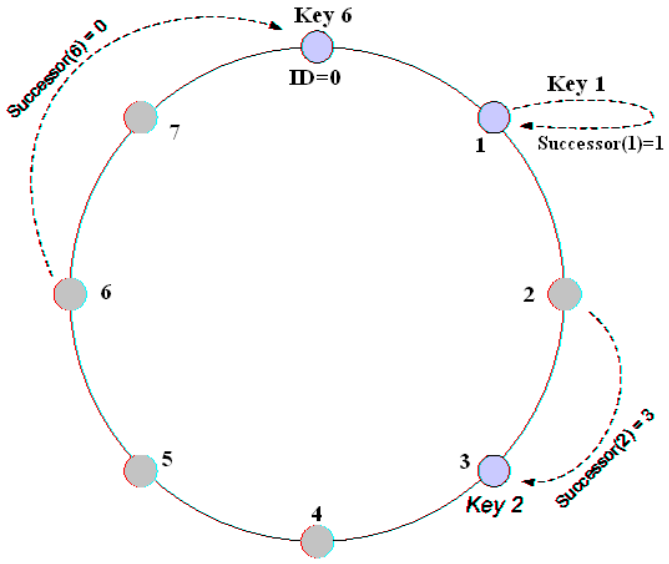


Figure 86 – Chord identifier circle (m=3) (adapted from [2])

Chord nodes employ a stabilization protocol that periodically updates the successor pointers and the finger table. The correct operation of the Chord depends on the actualization of the routing data. To avoid the situation where a node fails and other peers are left without any way of locating their new successor, nodes maintain a successor list, which contains a number of the peer's first successors. When a node's successor does not respond, it simply contacts the next peer on its successor list.

The performance of a Chord network degrades slowly as routing information loses validity, (due to node adhesion and abandonment), and availability remains high only as long as nodes fail independently. Given the fact that Chord's topology does not take into account the underlying physical IP network topology, a small connectivity failure in the IP network may cause multiple, scattered link failures in the Chord overlay [2].

The previously described searching mechanism may in a worst case scenario require the traversal of all nodes in the overly. To increase this mechanism's efficiency, Chord nodes are also to maintain additional routing information, in the form of a "finger table". Each table entry *i* points to the successor of node *n*+2*i*. For a node *n* to perform a lookup for key *k*, it consults the table to determine the highest node *n*¢ whose ID is between *n* and *k*. If node n¢ exists, the procedure is repeated starting from node n¢. If no such node exists, the successor of *n* is returned [2].

### B.1.3 *Tapestry*

Tapestry is a peer-to-peer overlay routing infrastructure which permits location-independent routing of messages directly to nearby replicas of an object or service resorting only to localized resources, It implements a self-repairing, soft-state-based routing layer that allows the bypassing of failed routes and nodes and the rapid adaptation of communication topologies to circumstances.

Tapestry's architecture employs a variation of the Plaxton [190] distributed search technique, combined with extra mechanisms to ensure availability, scalability, and adaptation in the case of failures and attacks.

The Plaxton mesh is a distributed data structure that allows peers to locate objects and reach them with messages through an arbitrarily-sized overlay network employing small and constant sized routing maps [2].

In a Plaxton mesh the nodes may perform three different roles:

- Servers – which store data objects;
- Routers – which forward messages between nodes ;
- Clients – which originate requests for data objects and consume them.

| | Level 5 | Level 4 | Level 3 | Level 2 | Level 1 |
|---|---|---|---|---|---|
| Entry 0 | 7493 | x0493 | xx093 | xxx03 | xxxx0 |
| Entry 1 | 17493 | x1493 | xx193 | xxx13 | xxxx1 |
| Entry 2 | 27493 | x2493 | xx293 | xxx23 | xxxx2 |
| Entry 3 | 37493 | x3493 | xx393 | xxx33 | xxxx3 |
| Entry 4 | 47493 | x4493 | xx493 | xxx43 | xxxx4 |
| Entry 5 | 57493 | x5493 | xx593 | xxx53 | xxxx5 |
| Entry 6 | **67493** | x6493 | xx693 | xxx63 | xxxx6 |
| Entry 7 | 77493 | x7493 | xx793 | xxx73 | xxxx7 |
| Entry 8 | 87493 | x8493 | xx893 | xxx83 | xxxx8 |
| Entry 9 | 97493 | x9493 | xx993 | xxx93 | xxxx9 |

Figure 87 – Neighbour Map maintained by a Tapestry Node with ID 67493 (adapted from [2])

Each node maintains a neighbour map (as presented in Figure 87) composed by multiple levels. Each level $l$ contains pointers to nodes whose ID must be matched with $l$ digits, this way, each table entry is a pointer to the closest network node with an ID which matches the number in the neighbour map up to a digit position (the $l^{th}$ position).
For instance, the 4th table entry for the 2nd level identifies the closest node to node 67493 in network distance whose ID ends in 43 [2].

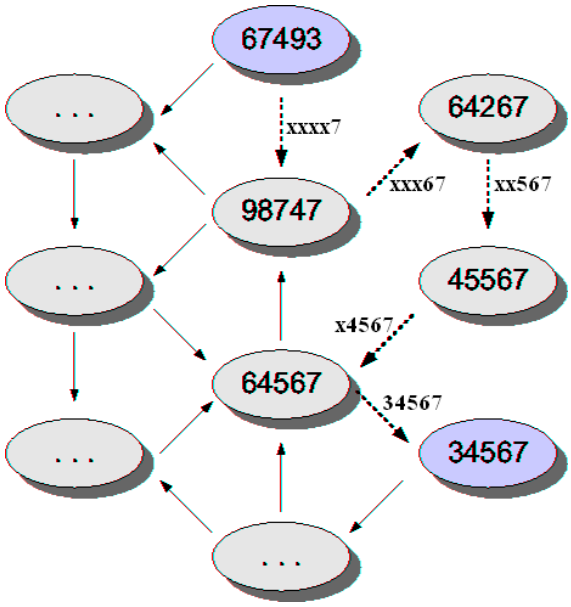


Figure 88 – Tapestry/Plaxton Mesh Routing Example using 5 digit long IDs (adapted from [2])

Within the Tapestry overlay, messages are routed to the destination peer, in a “digit by digit” manner, from the nodes on the right to the nodes on the left of the neighbour table (as presented in Figure 87).

An example path followed by a message from node with an ID value of 67493 to a node with an ID value of 34567 is presented in Figure 88. The end node location is reached by resolving (into the closest peer location) the digits of the desired node, one at a time at every node hop, as follows: xxxx7 → xxx67 → xx567 → x4567 → 34567.

The Plaxton scheme makes use of root nodes for the location of data objects, in order to assure that some node from which the object can be located is available.

Upon the object insertion of an object $o$ in the system and its storage at an $n_s$ node, a root node $n_r$ is attributed through the employment of a globally consistent deterministic algorithm. Then, $n_s$ issues a message to $n_r$, which triggers the storing at all nodes along the message's path, of data mapping the object id $o$ to storer id $n_s$.

At content location time, messages destined for $o$ are initially directed towards $n_r$, up to the point were a node is encountered which contains the ($o$, $n_s$) location mapping.

The Plaxton mesh's capability to route around a single link or node by selecting a node with a similar suffix provides the mesh with fault handling capabilities and the overall object location scheme provides it with scalability. On the other hand the need for a global or central knowledge for the attribution and identifying of root nodes is a possible point of failure for this system.

Whilst the Plaxton mesh assumes a static population of nodes, Tapestry augments this design so that it can handle the typically transient populations of peer-to-peer networks and provide adaptability and fault tolerance and in order to provide further optimizations. Some of the main additions brought on by Tapestry consist of the following [2]:

- Tapestry nodes maintain an additional list of back-pointers, which identifies nodes where they are pointed to as a neighbour. Such lists are employed in the procedures for the dynamic insertion of nodes, to allow the efficient creation of correct neighbour maps for new nodes;

- Tapestry adds semantic flexibility to the concept of distance between nodes. More than one object replica are stored in the system permitting the architecture to more flexibly define how the "closest" node will be determined;

- Cached content is maintained on a soft-state manner, based on an announce/listen approach. Tapestry employs this strategy to detect, circumvent and recover from failures in routing or object location. The caches maintained at the nodes are periodically updated by refreshment messages, or cleaned if no such messages are received. Furthermore the neighbour map is increased in order to maintain two backup neighbours in addition to the closest neighbour. When a node detects that a neighbour has for become unreachable, in order to avoid costly reinsertions, instead of removing the missing node's pointer, it temporarily flags it as invalid, so that it may have time to come back online if that is the case, and directs messages through alternative paths in the meantime;

- The root nodes constitute a single point of failure. In order to avoid this, Tapestry attributes multiple roots to each object. Tapestry employs surrogate routing to choose root peers incrementally, during the publishing process, to insert location information

into Tapestry. Surrogate allows the unique mapping of any identifier to an existing network node. The nodes selected to be the root or surrogate node for a data object is that which matches the data object's ID, *x*. Given the sparse nature of Tapestry's NodeID space the occurrence of such a node is unique. To verify the *x* node's existence a message is routed to it. If it does not exist, the routing of the message terminates when a neighbour map is reached, where the only non-empty routing entry belongs to its holding node. In such a case, that node is designated to be the surrogate root for the data object;

- Tapestry nodes optimize their neighbour pointers by periodically evaluating their network latency values. Algorithms are implemented to detect query hotspots and provide possible alternative locations for the storage of additional copies of objects so as to improve query response time. Nodes also maintain a "hotspot cache".

### B.1.4 *Kademlia*

Kademlia [191] is a peer-to-peer (key, value) pair storage and lookup system. This decentralized overlay network, much to the similarity of Pastry and Tapestry, employs the basic technique of attributing a NodeID to each peer, in a 160-bit key space, and of storing (key, value) tuples at the peers bearing IDs which are close to the key (data keys are also 160-bit identifiers). Within this system, tuple holding peers are located through a NodeID based routing algorithm which uses the keys as destination definers.

One of the fundamental innovations of Kademlia's architecture is its employment of an XOR metric for distance measuring between points in the key space. The symmetry of the XOR operation allows nodes to receive lookup queries from the same distribution of peers contained in their routing tables.
A Kademlia node may issue a query to any node within an interval, using latency as criteria to choose routes or send parallel asynchronous queries. It employs a single routing algorithm throughout the process to locate peers near a particular ID.
All messages transmitted by a Kademlia node include its NodeID. This allows the recipient node to apprehend the sender peer's existence [17].

The location of (key, value) pairs, by Kademlia nodes relies on the notion of distance between two identifiers. Such a distance is calculated as follows:

> The distance between two 160-bit identifiers, *a* and *b*, is defined as their bitwise exclusive OR (XOR), which is a non-Euclidean metric. XOR's unidirectional nature assures that for any specific point *x* and distance $d > 0$, there exists only one point *y* such that $d(x; y) = d$. This guarantees that all lookups for the same key converge along the same path, irrespectively of the originating peer. This way, caching (key, value) pairs throughout the lookup path alleviates hot spots.

A Kademlia node stores a list of (IP address, UDP port, NodeID) tuples for peers within a certain distance. These lists (or *k*-buckets) are kept ordered by last time contacted, (least recently accessed node at the head, most recently accessed at the tail).

The routing protocol employed by Kademlia is composed by:

- PING messages which are used to probe a node in order to determine if it is active;
- STORE instructs a node to store a (key, value) pair for posterior retrieval;
- FIND NODE receives a 160-bit ID, and returns a {IP address, UDP port, NodeID} tuple for a number of the peers it knows that are closest to the target ID;
- FIND VALUE is similar to FIND NODE, it returns {IP address, UDP port, NodeID} tuples, except for the case where a node received a STORE for the key. In such a case, the stored value is returned.

To locate a (key, value) pair, Kademlia nodes commence by performing a FIND VALUE lookup to find the *k* peers with IDs closest to the pair's key. To adhere to the network, a peer *n* must know an already participating peer *m*. Peer *n* adds peer *m* to the appropriate *k*-bucket, and then undertakes a peer lookup for its own NodeID. This procedure refreshes all *k*-buckets further away than *n*'s closest neighbour, and *n* also fills its own *k*-buckets and adds itself to other nodes' *k*-buckets [17].

# Annex C – BM Inquiry Results

## C.1 Introduction

The inquiry was composed of four parts. Said parts and their purposes are the following:

- Query group 1 – its purpose is to:
  - evaluate user acceptance of correctness of rewarding content producers and CDists;
  - evaluate user awareness of the fact that if such rewarding does not take place the content will cease to be produced;

- Query group 2 – its purpose is to evaluate user receptivity to a total dematerialization of content distribution (i.e. distribution performed completely on-line);

- Query group 3 – its purpose is to assess user general receptivity/preference relatively to content accessing modes supported though a lateral extraction of gains in opposition to those which imply a direct and mandatory payment for content access;

- Query group 4 – its purpose is to evaluate user receptivity to three specific content accessing modes based on lateral gain extraction. As such it:
  - evaluates user receptivity to an open access mode, supported by voluntary donations;
  - evaluates user receptivity to an open access mode, supported by advertisement viewing;
  - evaluates user receptivity to an open access mode, supported by ransoms.

## C.2 Results

The specific queries that where defined, and the obtained response results are presented below, discriminated by query group (results presented in percentage terms).

### C.2.1 *Query Group 1*

| Query/Answer | I completely agree | I agree | I neither agree nor disagree | I disagree | I completely disagree |
|---|---|---|---|---|---|
| **1.1 It is fair that media content producers are rewarded for the content that they produce** | **59** | **39** | **2** | **0** | **0** |
| **1.2 It is fair that the intermediary players, in the field of media content production, are rewarded for their work, in the support of the action of media content producers.** | 23 | **63** | 11 | 1 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| **1.3**<br>**The continued production of the media goods (movies, series, music, etc.), that we consume on a daily basis, depends on the economical compensation of that activity. If the final consumers stop compensating said activity those goods will cease to be produced.** | **20** | **45** | **15** | **12** | **9** |
| **1.4**<br>**It is fair that I systematically access and consume media goods without compensating, in any way, the producers of such goods.** | **2** | **15** | **27** | **48** | **9** |

## C.2.2 *Query Group 2*

| Query/Answer | I completely agree | I agree | I neither agree nor disagree | I disagree | I completely disagree |
|---|---|---|---|---|---|
| **2.1**<br>**For the same payment modes, on-line media goods delivery is more convenient (to me as a consumer), and more efficient than the traditional distribution of such goods.** | **28** | **45** | **16** | **10** | **1** |
| **2.2**<br>**It is acceptable that media goods are to be predominately distributed on-line, and that physical distributions of such goods (based on CDs, DVDs, etc.), are restricted to "vintage" or "collector" editions.** | **9** | **33** | **24** | **24** | **10** |

## C.2.3 *Query Group 3*

| Query/Answer | I completely agree | I agree | I neither agree nor disagree | I disagree | I completely disagree |
|---|---|---|---|---|---|
| **3.1**<br>**The models of media content access on-line should be based on indirect and/or voluntary payment instead of on direct and mandatory one.** | **17** | **44** | **30** | **9** | **0** |
| **3.3**<br>**I prefer an on-line media content access and producer rewarding model that allows me to have some control of how the revenue is shared, (amongst producers and others), to a model where I have no such power or responsibility.** | **18** | **46** | **29** | **5** | **1** |

| | A mode where you had to pay for content access, where such payment would liberate only your access to that content and said access would be subjected to some constraints (e.g. the number of devices where you could consume the content). | A mode where you made only voluntary donations to the artists of your preference, and where the access to media content was free, both for you and everyone else, regardless of any contribution (donation), yours or theirs. |
|---|---|---|
| **3.2 Considering that you would spend about the same monthly amount, which content access mode would you prefer?** | **30** | **70** |

## C.2.4 *Query Group 4*

| Query/Answer | I completely agree | I agree | I neither agree nor disagree | I disagree | I completely disagree |
|---|---|---|---|---|---|
| **4.1 If I had convenient and secure ways to do it, I would voluntarily perform monetary donations to artists and content producers that were responsible for the creation of media products (movies, songs, etc), which I considered worthy of being rewarded.** | **13** | **49** | **22** | **13** | **2** |
| **4.3 The compensation of media content producers, by way of voluntary donations, enables that such entities are rewarded in a fairer manner.** | **10** | **38** | **35** | **13** | **4** |
| **4.6 I am willing to watch commercial advertisements in order to contribute (with my attention, which is being resold to advertisers), to the rewarding of artists/content producers.** | **21** | **52** | **15** | **11** | **1** |
| **4.8 If my "consumption" of advertisement messages is rewarded with some form of "credits" that I could donate to artists of my preference, I would voluntarily request the viewing of such messages.** | **24** | **54** | **18** | **4** | **0** |
| **4.10 Within the context of the donation based rewarding of artists and media producers, if one such entity, that I trusted, requested to be financed, in advance, for the production of a media good (that I considered that I was going to like), I would have no problems in donating some amount to him, even before the actual good existed.** | **11** | **32** | **22** | **24** | **11** |

| | Yes | No |
|---|---|---|
| **4.2 Have you ever performed any monetary donation to some on-line initiative?** | **17** | **83** |

| | 1 to 5 € | 5 to 10 € | 10 to 20 € | 20 to 50 € | more than 50 € |
|---|---|---|---|---|---|
| **4.4 Considering that you would not have to spend any other resources to acquire the media goods of your preference, other than the amounts that you donated, how much would you be willing to donate (in average), on a monthly basis, to the set of artists and content producers of your preference?** | **56** | **33** | **11** | **0** | **0** |

| | 1 to 5 € | 5 to 10 € | 10 to 20 € | 20 to 50 € | more than 50 € |
|---|---|---|---|---|---|
| **4.5 And how much would you be willing to donate to each individual artist, in the same conditions ant time interval?** | **87** | **13** | **0** | **0** | **0** |

| | A model where publicity is juxtaposed with the media contet and must be consumed, for instance, before you can listen to/watch the desired content (for instance, in YouTube, before you can watch some of the videos you must first watch an advertisement). | A model where the advertisement messages are separated from the media content and where you have to be the one to request the viewing of such messages, and thus, where you may "consume" them at the moment of your preference. |
|---|---|---|
| **4.7 The two ways, of supporting content producers' activity, presented below, are based on advertisement. Considering that you would have to "consume" the same amount of advertising messages in both of them, which one would you prefer?** | **43** | **57** |

| | 1 to 5 | 5 to 10 | 10 to 20 | 20 to 30 | more than 30 |
|---|---|---|---|---|---|
| **4.9 In the context of question 4.8, how many minutes of your attention would you be willing to donate, on a daily basis, through the viewing of advertisement messages?** | **51** | **38** | **10** | **0** | **1** |

# Annex D – Data Model Info

## D.1 Registered Information

### D.1.1 *Introduction*

The registry of all system events will comprise the following basic set of data fields:

- The event's ID – the system-wide unique identifier of the event;
- The event's beginning time – the time at which the event begun its existence, i.e. its participation in the system;
- The event's end time – the time at which the event ceased its existence, i.e. its participation in the system;
- The EntCompEv's validity state – a Boolean field indicating if the event is still existent;

The following sections describe all the registered system events indicating the connections that bind all such registries into a coherent and meaningful tissue.

### D.1.2 *Registered Global Information*

**D.1.2.1** *Registered Global PLL Information*

**D.1.2.1.1** *Registered Global PLL Relational and Procedural Complex Events*

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***Peer Registration Event*** | | |
| | Event Nr | #1 | | |
| | Explanation | This is an event that represents a peer's registration with the system. | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | *Peer-participationIn-System* | 1 |
| | | *Event-casusationOf-Event* | *PregCertif -pertinenceTo- Peer-participationIn-System* | 1 |
| | | *Event-initiationOf-Event* | *Peer-maintenanceOf-PeerIP* | 1 |
| | | *Event-initiationOf-Event* | *Peer-maintenanceOf-PeerKeyPair* | 1 |
| | | *Event-initiationOf-Event* | *User-ownageOf-Peer* | 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***Peer Registration Update Event*** | | |
| | Event Nr | #2 | | |
| | Explanation | This is an event that represents a peer's updating of its registration with the system. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *PLoginCertif-pertinenceTo-Peer* | 0 –1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-casusationOf-Event* | *PregCertif -pertinenceTo- Peer-participationIn-System* | 1 |
| | | *Event-terminationOf-Event* | *Peer-maintenanceOf-PeerKeyPair* | 1 |
| | | *Event-initiationOf-Event* | *Peer-maintenanceOf-PeerKeyPair* | 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***Peer Deregistration Event*** | | |
| | Event Nr | #3 | | |
| | Explanation | This is an event that represents a peer's registration with the system. | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *Peer-participationIn-System* | 1 |
| | | *Event-terminationOf-Event* | *Peer-maintenanceOf-PeerIP* | 1 |
| | | *Event-terminationOf-Event* | *Peer-maintenanceOf-PeerKeyPair* | 1 |
| | | *Event-terminationOf-Event* | *User-ownageOf-Peer* | 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***Peers Evaluation Event*** | | |
| | Event Nr | #4 | | |
| | Explanation | This is a periodical, internal, system event which consists of an evaluation of all of the system peer's conduit. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***Peer Evaluation Event*** | 1 – ∞ |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***Peer Evaluation Event*** | | |
| | Event Nr | #5 | | |
| | Explanation | This is an internal system event. It consists of the evaluation of a specific peers's conduit, taking into consideration other peers' misbehaviour reports regarding that peer. It may lead to the expulsion, quarantining or dequarantining of the evaluated peer. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *System-evaluationOf-Peer* | 1 |
| | | *Event-inclusionOf-Event* | *System-analysisOf-PeerMissbehavReport* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *(System-analysisOf-PeerMissbehavReport)-pertainmentTo-(System-evaluationOf-Peer)* | 0 – ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | *System-evictionOf-Peer* | 0 – 1 |
| | | *Event-initiationOf-Event* | *System-quarantiningOf-Peer* | 0 – 1 |
| | | *Event-terminationOf-Event* | *System-quarantiningOf-Peer* | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| Rel Ev | Event Name | ***System-evictionOf-Peer*** | | |
| | Event Nr | #6 | | |
| | Explanation | This is a relational event that represents a peer's eviction from the system. | | |
| | Subject Event | *System* | | |
| | Object Event | *Peer* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***User Disconnection Event*** | 0 - ∞ |
| | | *Event-causationOf-Event* | ***Peer Disconnection Event*** | 0 –1 |
| | | *Event-causationOf-Event* | ***MO Deregistration event*** | 0 - ∞ |
| | | *Event-causationOf-Event* | ***Peer Deregistration event*** | 0 - 1 |

| Rel Ev | Event Name | ***Peer-loginTo-System*** | | |
|---|---|---|---|---|
| | Event Nr | #7 | | |
| | Explanation | This is a relational event that represents a peers's action to log into the system. | | |
| | Subject Event | *Peer* | | |
| | Object Event | *System* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***Peer Connection Event*** | 0 –1 |

| Rel Ev | Event Name | ***Peer-logoffFrom-System*** | | |
|---|---|---|---|---|
| | Event Nr | #8 | | |
| | Explanation | This is a relational event that represents a peer's action to log out of the system. | | |
| | Subject Event | *Peer* | | |
| | Object Event | *System* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***Peer Disconnection Event*** | 0 –1 |

| Proc Comp Ev | Event Name | ***Peer Connection Event*** | | |
|---|---|---|---|---|
| | Event Nr | #9 | | |
| | Explanation | This is an event which represents system procedures to perform a peers's connection (logging in) to the system. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *PLoginCertif-pertinenceTo-Peer* | 0 –1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | ***Peer Participation Session Event*** | 0 –1 |
| | | *Event-initiationOf-Event* | ***PLL Servicing Session Event*** | 1 |
| | | *Event-initiationOf-Event* | ***UEL Servicing Session Event*** | 1 |

| Proc Comp Ev | Event Name | ***Peer Disconnection Event*** | | |
|---|---|---|---|---|
| | Event Nr | #10 | | |
| | Explanation | This is an event which represents system procedures to perform a peer's disconnection (logoff) to the system. | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | ***Peer Participation Session Event*** | 0 –1 |
| | | *Event-terminationOf-Event* | ***PLL Servicing Session Event*** | 1 |
| | | *Event-terminationOf-Event* | ***UEL Servicing Session Event*** | 1 |

| Proc Comp Ev | Event Name | ***Peer Participation Session Event*** | | |
|---|---|---|---|---|
| | Event Nr | #11 | | |
| | Explanation | This event consists of a peer participation session in the system. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***Peer Reporting Event*** | 0 - ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-supportingOf-Event* | ***User Attending Session Event*** | 0 – ∞ |

| Proc Comp Ev | Event Name | ***Peer Reporting Event*** | | |
|---|---|---|---|---|
| | Event Nr | #12 | | |
| | Explanation | This event consists of a peer reporting the misbehaviour of another peer to the system. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *Peer-denouncingOf-Peer* | 1 |
| | | *Event-inclusionOf-Event* | *PMissbehavReport -pertinenceTo- (Peer-denouncingOf-Peer)* | 1 |

| Proc Comp Ev | Event Name | ***PLL Servicing Session Event*** | | |
|---|---|---|---|---|
| | Event Nr | #13 | | |
| | Explanation | This event consists of the maintenance of PLL servicing responsabilities (over the peripheral peer collective) by a specific OCP. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *(Peer(OCP) -pllServicingOf- Peer(PP))* | 1 |
| | | *Event-inclusionOf-Event* | *Instruction OP IO -pertainmentTo- (Peer(OCP) -pllServicingOf- Peer(PP))* | 1 |

**D.1.2.1.2** *Registered Global PLL Entitary Complex Events*

| Ent Comp Ev | Event Name | ***Peer*** | |
|---|---|---|---|
| | Event Nr | #14 | |
| | Explanation | This event consists of a peer. | |
| | Data Fields | *Peer Type* | The peer type (PP, OCP, or CCP) |
| | | *Peer Pub Key* | The peer's public key |
| | | *Peer Priv Key* | The peer's private key |
| | | *Is Quarantined* | An indication if the peer is quarantined |

| Ent Comp Ev | Event Name | ***Peer Key Pair*** | |
|---|---|---|---|
| | Event Nr | #15 | |
| | Explanation | This event consists of a peer's public and private key pair. | |
| | | *Pub Key* | The peer's public key |
| | | *Priv Key* | The peer's private key |

### **D.1.2.2** *Registered Global UEL Information*

#### **D.1.2.2.1** *Registered Global UEL Relational and Procedural Complex Events*

##### **D.1.2.2.1.1** *User Originated*

| Rel Ev | Event Name | ***User-joiningOf-System*** | | |
|---|---|---|---|---|
| | Event Nr | #16 | | |
| | Explanation | This is a relational event that represents a user's action to join the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *System* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***User Registration Event*** | 0 - 1 |

| Rel Ev | Event Name | ***User-abandoningOf-System*** | | |
|---|---|---|---|---|
| | Event Nr | #17 | | |
| | Explanation | This is a relational event that represents a user's action to leave the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *System* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***User Derigistration Event*** | 1 |

| Proc Comp Ev | Event Name | ***User Registration Event*** | | |
|---|---|---|---|---|
| | Event Nr | #18 | | |
| | Explanation | This is an event which represents system procedures to perform a user registration. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-ownageOf-UAccount* | 1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | *User-participationIn-System* | 1 |
| | | *Event-initiationOf-Event* | *User-maintenanceOf-UserKeyPair* | 1 |
| | | *Event-casusationOf-Event* | *URegCertif -pertinenceTo- User-participationIn-System* | 1 |

| Proc Comp Ev | Event Name | ***User Registration Update Event*** | | |
|---|---|---|---|---|
| | Event Nr | #19 | | |
| | Explanation | This is an event which represents system procedures to perform a user registration update. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *ULoginCertif-pertinenceTo-User* | 0 –1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *User-maintenanceOf-UserKeyPair* | 1 |
| | | *Event-initiationOf-Event* | *User-maintenanceOf-UserKeyPair* | 1 |
| | | *Event-casusationOf-Event* | *URegCertif -pertinenceTo- User-participationIn-System* | 1 |

| Proc Comp Ev | Event Name | ***User Deregistration Event*** | | |
|---|---|---|---|---|
| | Event Nr | #20 | | |
| | Explanation | This is an event which represents system procedures to perform a user deregistration. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***User Disconnection Event*** | 0 - 1 |
| | | *Event-inclusionOf-Event* | ***MO Deregistration Event*** | 0 - 1 |
| | | *Event-inclusionOf-Event* | ***Peer Deregistration Event*** | 0 - 1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *User-participationIn-System* | 1 |
| | | *Event-terminationOf-Event* | *User-maintenanceOf-UserKeyPair* | 1 |

| Rel Ev | Event Name | ***User-loginTo-System*** | | |
|---|---|---|---|---|
| | Event Nr | #21 | | |
| | Explanation | This is a relational event that represents a user's action to log into the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *System* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***User Connection Event*** | 0 –1 |

| Rel Ev | Event Name | ***User-logoffFrom-System*** | | |
|---|---|---|---|---|
| | Event Nr | #22 | | |
| | Explanation | This is a relational event that represents a user's action to log out of the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *System* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***User Disconnection Event*** | 0 –1 |

| Proc Comp Ev | Event Name | ***User Connection Event*** | | |
|---|---|---|---|---|
| | Event Nr | #23 | | |
| | Explanation | This is an event which represents system procedures to perform a user's connection (logging in) to the system. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *ULoginCertif-pertinenceTo-User* | 0 –1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | ***User Attending Session Event*** | 0 –1 |

| Proc Comp Ev | Event Name | ***User Disconnection Event*** | | |
|---|---|---|---|---|
| | Event Nr | #24 | | |
| | Explanation | This is an event which represents system procedures to perform a user's disconnection (logoff) to the system. | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | ***User Attending Session Event*** | 0 –1 |

| Pro Comp Ev | Event Name | ***User Attending Session Event*** | | |
|---|---|---|---|---|
| | Event Nr | #25 | | |
| | Explanation | This event consists of a user participation session in the system. It starts with a user login and ends with his logoff. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-loginTo-System* | 1 |
| | | *Event-inclusionOf-Event* | *User-logoffFrom-System* | 0 – 1 |
| | | *Event-inclusionOf-Event* | *Forceful User Logout* | 0 – 1 |
| | | *Event-inclusionOf-Event* | ***User Registration Update Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *User-InsertionOf-MO* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *User-removalOf-MO* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***Add MO Insertion Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***Add MO Removal Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *User-insertionOf-MORansmAnnoncement* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *User-removalOf-MORansmAnnoncement* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***MO Search Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *User-consumptionOf-Mo* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *User-consumptionOf-MORansmAnnoncement* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***MO Ownership Exchange Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***MO Commenting Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***MO Responding Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***MO Versioning Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***User Reporting Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***Donation Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***User Attention Sale Event*** | 0 – ∞ |

| | | | | |
|---|---|---|---|---|
| | | *Event-inclusionOf-Event* | *User-loadingOf-UAccount* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *User-unloadingOf-UAccount* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***Peer Ownership Exchange*** | 0 – ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-precedenceOf-Event* | ***User Attending Session Event*** | 0 – ∞ |

| | | | | |
|---|---|---|---|---|
| Rel Ev | Event Name | ***User-insertionOf-MO*** | | |
| | Event Nr | #26 | | |
| | Explanation | This is a relational event which represents the user initiated action of inserting an MO into the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *MO* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***MO Registration Event*** | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***MO Registration Event*** | | |
| | Event Nr | #27 | | |
| | Explanation | This is an event which represents system procedures to perform an MO registration. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | *User-ownageOf-MO* | 1 |
| | | *Event-initiationOf-Event* | *SemanticConcept -characterizationOf- MO* | 1 – ∞ |
| | | *Event-initiationOf-Event* | *MO -pertainmentTo- MORnsmAnn* | 0 – ∞ |
| | | *Event-initiationOf-Event* | ***MO Storage Event*** | 1 – ∞ |

| | | | | |
|---|---|---|---|---|
| Rel Ev | Event Name | ***User-removalOf-MO*** | | |
| | Event Nr | #28 | | |
| | Explanation | This is a relational event which represents the user initiated action of removing an MO from the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *MO* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***MO Deregistration Event*** | 0 - 1 |

| Proc Comp Ev | Event Name | ***MO Deregistration Event*** | | |
|---|---|---|---|---|
| | Event Nr | #29 | | |
| | Explanation | This is an event which represents system procedures to perform an MO deregistration. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *User-ownageOf-MO* | 1 |
| | | *Event-terminationOf-Event* | *SemanticConcept -characterizationOf- MO* | 1 – ∞ |
| | | *Event-terminationOf-Event* | *MO -pertainmentTo- MORnsmAnn* | 0 – ∞ |
| | | *Event-terminationOf-Event* | *MO-commentingTo-MO* | 0 – ∞ |
| | | *Event-terminationOf-Event* | *MO-versioningOf-MO* | 0 – ∞ |
| | | *Event-terminationOf-Event* | *MO-respondingTo-MO* | 0 – ∞ |
| | | *Event-terminationOf-Event* | *System-quarantiningOf-MO* | 0 – 1 |
| | | *Event-terminationOf-Event* | ***MO Storage Event*** | 1 – ∞ |

| Proc Comp Ev | Event Name | ***Add MO Insertion Event*** | | |
|---|---|---|---|---|
| | Event Nr | #30 | | |
| | Explanation | This is a user initiated event that consists of the insertion of an Advertisement MO into the system. | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-insertionOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *User-insertionOf-InquiryIO* | 1 – ∞ |
| | | *Event-inclusionOf-Event* | *(User-insertionOf-InquiryIO) -pertainmentTo- (User-insertionOf-MO)* | 1 – ∞ |
| | | *Event-inclusionOf-Event* | *User-insertionOf-InquiryRespIO* | 1 – ∞ |
| | | *Event-inclusionOf-Event* | *(User-insertionOf-InquiryRespIO) -pertainmentTo- (User-insertionOf-MO)* | 1 – ∞ |

| Proc Comp Ev | Event Name | ***Add MO Removal Event*** | | |
|---|---|---|---|---|
| | Event Nr | #31 | | |
| | Explanation | This is a user initiated event that consists of the removal of an MO from the system. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-removalOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *User-removalOf-InquiryIO* | 1 – ∞ |
| | | *Event-inclusionOf-Event* | *(User-removalOf-InquiryIO) -pertainmentTo- (User-removalOf-MO)* | 1 – ∞ |
| | | *Event-inclusionOf-Event* | *User-removalOf-InquiryRespIO* | 1 – ∞ |
| | | *Event-inclusionOf-Event* | *(User-removalOf-InquiryRespIO)* | 1 – ∞ |

| | | | | |
|---|---|---|---|---|
| | | | *-pertainmentTo-*<br>*(User-removalOf-MO)* | |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *System-quarantiningOf-MO* | 0 – 1 |

| Rel Ev | | | | |
|---|---|---|---|---|
| | Event Name | ***User-insertionOf-InquiryIO*** | | |
| | Event Nr | #32 | | |
| | Explanation | This event consists of the insertion of an Inquiry IO into the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *Inquiry IO* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***OpIO Registration Event*** | 0 – 1 |

| Rel Ev | | | | |
|---|---|---|---|---|
| | Event Name | ***User-insertionOf-InquiryRespIO*** | | |
| | Event Nr | #33 | | |
| | Explanation | This event consists of the insertion of an Inquiry Response IO into the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *InquiryRespIO* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***OpIO Registration Event*** | 0 – 1 |

| Rel Ev | | | | |
|---|---|---|---|---|
| | Event Name | ***User-removalOf-InquiryIO*** | | |
| | Event Nr | #34 | | |
| | Explanation | This event consists of the removal of an Inquiry IO into the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *InquiryRespIO* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***OpIO Deregistration Event*** | 0 – 1 |

| Rel Ev | | | | |
|---|---|---|---|---|
| | Event Name | ***User-removalOf-InquiryRespIO*** | | |
| | Event Nr | #35 | | |
| | Explanation | This event consists of the removal of an Inquiry Response IO into the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *InquiryRespIO* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***OpIO Deregistration Event*** | 0 – 1 |

| Proc Comp Ev | Event Name | ***OpIO Registration Event*** | | |
|---|---|---|---|---|
| | Event Nr | #36 | | |
| | Explanation | This is an event which represents system procedures to perform the registration of an operational IO. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | *OpIO-pertainmentTo-MO* | 0 – ∞ |
| | | *Event-initiationOf-Event* | *OpIO-pertainmentTo-Peer* | 0 – ∞ |
| | | *Event-initiationOf-Event* | *OpIO-pertainmentTo-User* | 0 – ∞ |
| | | *Event-initiationOf-Event* | *OpIO-pertainmentTo-OpIO* | 0 – ∞ |
| | | *Event-initiationOf-Event* | *Peer-storageOf-OpIO* | 0 – ∞ |

| Proc Comp Ev | Event Name | ***OpIO Deregistration Event*** | | |
|---|---|---|---|---|
| | Event Nr | #37 | | |
| | Explanation | This is an event which represents system procedures to perform the deregistration of an operational IO. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *OpIO-pertainmentTo-MO* | 0 – ∞ |
| | | *Event-terminationOf-Event* | *OpIO-pertainmentTo-Peer* | 0 – ∞ |
| | | *Event-terminationOf-Event* | *OpIO-pertainmentTo-User* | 0 – ∞ |
| | | *Event-terminationOf-Event* | *OpIO-pertainmentTo-OpIO* | 0 – ∞ |
| | | *Event-terminationOf-Event* | *Peer-storageOf-OpIO* | 0 – ∞ |

| Rel Ev | Event Name | ***Peer-storageOf-OpIO*** |
|---|---|---|
| | Event Nr | #38 |
| | Explanation | This is a relational event that represents the storage of an operational IO by some specific peer. |
| | Subject Event | *Peer* |
| | Object Event | *All types of operational events* |

| Rel Ev | Event Name | ***OpIO-pertainmentTo-OpIO*** |
|---|---|---|
| | Event Nr | #39 |
| | Explanation | This is a relational event that represents a logical relationship between operational IOs. |
| | Subject Event | *InquiryIO, InquiryRespIO* |
| | Object Event | *InquiryIO, InquiryRespIO* |

| Rel Ev | Event Name | ***User-insertionOf-MORansomAnnoncement*** |
|---|---|---|
| | Event Nr | #40 |
| | Explanation | This is a user initiated event that consists of the insertion of an MO Ransom Announcement into the system. |
| | Subject Event | *User* |

| | | | | |
|---|---|---|---|---|
| | Object Event | *MORansomAnnoncement* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***MORansomAnnoncement Registration Event*** | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| Rel Ev | Event Name | ***User-removalOf-MORansomAnnoncement*** | | |
| | Event Nr | #41 | | |
| | Explanation | This is a user initiated event that consists of the removal of an MO Ransom Announcement from the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *MORansomAnnoncement* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***MORansomAnnoncement Deregistration Event*** | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***MORansomAnnoncement Registration Event*** | | |
| | Event Nr | #42 | | |
| | Explanation | This is an event which represents system procedures to perform an MORansomAnnoncement registration. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | *User-ownageOf-MORnsmAnn* | 1 |
| | | *Event-initiationOf-Event* | *SemanticConcept -characterizationOf- MORnsmAnn* | 1 – ∞ |
| | | *Event-initiationOf-Event* | *Peer -storageOf- MORansomAnnoncement* | 1 – ∞ |
| | | *Event-initiationOf-Event* | ***Ransoming Event*** | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***MORansomAnnoncement Deregistration Event*** | | |
| | Event Nr | #43 | | |
| | Explanation | This is an event which represents system procedures to perform an MORansomAnnoncement deregistration. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *User-ownageOf-MORnsmAnn* | 1 |
| | | *Event-terminationOf-Event* | *SemanticConcept -characterizationOf- MORnsmAnn* | 1 – ∞ |
| | | *Event-terminationOf-Event* | *Peer -storageOf- MORansomAnnoncement* | 1 – ∞ |
| | | *Event-terminationOf-Event* | *Ransoming Event* | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***MO Search Event*** | | |
| | Event Nr | #44 | | |
| | Explanation | This event consists of the performing, by a user, of a search for MOs, based on user specified search criteria. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-emissionOf-SearchQuery* | 1 |
| | | *Event-inclusionOf-Event* | *(User-emissionOf-SearchQuery) -causationOf- (System-returningOf-SQRO)* | 1 |
| | | *Event-inclusionOf-Event* | *SQRO-respondingTo-SearchQuery* | 1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | ***OpIO(SQRO) Registration Event*** | 1 |

| | | | | |
|---|---|---|---|---|
| Rel Ev | Event Name | ***User-consumptionOf-MO*** | | |
| | Event Nr | #45 | | |
| | Explanation | This event consists of the consumption, by a user, of a specific MO. | | |
| | Subject Event | *User* | | |
| | Object Event | *MO* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | ***MO Storage Event*** | 1 |
| | | *Event-initiationOf-Event* | ***OpIO(SQRO) Deregistration Event*** | 0 - 1 |

| | | | | |
|---|---|---|---|---|
| Rel Ev | Event Name | ***User-consumptionOf-MORansomAnnoncement*** | | |
| | Event Nr | #46 | | |
| | Explanation | This event consists of the consumption, by a user, of a specific MORansomAnnoncement. | | |
| | Subject Event | *User* | | |
| | Object Event | *MORansomAnnoncement* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | *Peer -storageOf- MORansomAnnoncement* | 1 |

| Proc Comp Ev | Event Name | ***MO Storage Event*** | | |
|---|---|---|---|---|
| | Event Nr | #47 | | |
| | Explanation | This event consists of the storage, by a specific peer, of a specific MO. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *Peer-StorageOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *(Peer-StorageOf-MO) -implimentOf- (Peer-storageOf-MOFrag)* | 1 – ∞ |
| | | *Event-inclusionOf-Event* | *Peer-storageOf-MOFrag* | 1 – ∞ |

| Proc Comp Ev | Event Name | ***MO Ownership Exchange Event*** | | |
|---|---|---|---|---|
| | Event Nr | #48 | | |
| | Explanation | This event consists of the exchanging of ownership, over an MO, between two users. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User -proposalOf- (User-ownageOf-MO)* | 1 |
| | | *Event-inclusionOf-Event* | *User -acceptanceOf- (User-ownageOf-MO)* | 0 – 1 |
| | | *Event-inclusionOf-Event* | *User -rejectionOf- (User-ownageOf-MO)* | 0 – 1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *User-ownageOf-MO* | 0 – 1 |
| | | *Event-initiationOf-Event* | *User-ownageOf-MO* | 0 – 1 |

| Proc Comp Ev | Event Name | ***MO Commenting Event*** | | |
|---|---|---|---|---|
| | Event Nr | #49 | | |
| | Explanation | This event consists of a user inserting an MO into the system which consists of a comment to some other MO. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-insertionOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *(User-insertionOf-MO) -establishingOf- (MO-commentingTo-MO)* | 1 |

| Proc Comp Ev | Event Name | ***MO Responding Event*** | | |
|---|---|---|---|---|
| | Event Nr | #50 | | |
| | Explanation | This event consists of a user inserting an MO into the system which consists of a response to some other MO. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-insertionOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *(User-insertionOf-MO)-establishingOf-(MO-respondingTo-MO)* | 1 |

| Proc Comp Ev | Event Name | ***MO Versioning Event*** | | |
|---|---|---|---|---|
| | Event Nr | #51 | | |
| | Explanation | This event consists of a user inserting an MO into the system which consists of a different version of some other MO (of those owned by the inserting user). | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-insertionOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *(User-insertionOf-MO)-establishingOf-(MO-versioningOf-MO)* | 1 |

| Proc Comp Ev | Event Name | ***User Reporting Event*** | | |
|---|---|---|---|---|
| | Event Nr | #52 | | |
| | Explanation | This event consists of a user reporting the misbehaviour of another user (typically) regarding MO inserted by the latter. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-denouncingOf-User* | 1 |
| | | *Event-inclusionOf-Event* | *UMissbehavReport-pertinenceTo-(User-denouncingOf-User)* | 1 |

| Proc Comp Ev | Event Name | ***Peer Ownership Exchange Event*** | | |
|---|---|---|---|---|
| | Event Nr | #53 | | |
| | Explanation | This event consists of the exchanging of ownership, over a Peer, between two users. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-proposalOf-(User-ownageOf-Peer)* | 1 |
| | | *Event-inclusionOf-Event* | *User-acceptanceOf-(User-ownageOf-Peer)* | 0 – 1 |
| | | *Event-inclusionOf-Event* | *User-rejectionOf-* | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| | | | *(User-ownageOf-Peer)* | |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | *User-ownageOf-Peer* | 0 – 1 |
| | | *Event-initiationOf-Event* | *User-ownageOf-Peer* | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***Donation Event*** | | |
| | Event Nr | #54 | | |
| | Explanation | This event consists of a user donating some monetary resources to another user, on the grounds of some MO produced by the latter, that the earlier deems meritorious of a reward. | | |
| | Data Fields | *Total Donated Amount* | The value of the total amount donated by the donating user | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-likingOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *(User-likingOf-MO) -causationOf- (User-rewardingOf-User)* | 1 |
| | | *Event-inclusionOf-Event* | *User-rewardingOf-User* | 1 |
| | | *Event-inclusionOf-Event* | *System -taxatingOf- (User-rewardingOf-User)* | 1 |

| | | | |
|---|---|---|---|
| Rel Ev | Event Name | ***User-rewardingOf-User*** | |
| | Event Nr | #55 | |
| | Explanation | This is a relational event that represents a user action which consists of rewarding another user with a specific amount of monetary resources. | |
| | Subject Event | *User* | |
| | Object Event | *User* | |
| | Data Fields | *Received Amount* | The amount received by the rewarded user. |

| | | | |
|---|---|---|---|
| Rel Ev | Event Name | ***System-taxationOf-(User-rewardingOf-User)*** | |
| | Event Nr | #56 | |
| | Explanation | This is a relational event that represents the system's action of taxing an inter-user donation. | |
| | Subject Event | *User* | |
| | Object Event | *User* | |
| | Data Fields | *Taxed Amount* | The amount extracted by the system as tax. |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***User Attention Sale Event*** | | |
| | Event Nr | #57 | | |
| | Explanation | This event consists of the sale, by a user, of his attention, to another user (advertiser), by viewing some specific MO containing an advertisement message. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-consumptionOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *User-answeringOf-InquiryIO* | 1 |
| | | *Event-inclusionOf-Event* | *User-paymentTo-User* | 0 – 1 |
| | | *Event-inclusionOf-Event* | *System -taxatingOf- (User-paymentTo-User)* | 0 – 1 |

| | | | |
|---|---|---|---|
| Rel Ev | Event Name | ***User-paymentTo-User*** | |
| | Event Nr | #58 | |
| | Explanation | This is a relational event that represents a user action which consists of paying another user with a specific amount of monetary resources, for attention that the latter sold to the earlier. | |
| | Subject Event | *User* | |
| | Object Event | *User* | |
| | Data Fields | *Paid Amount* | The amount paid by the advertiser user. |

| | | | |
|---|---|---|---|
| Rel Ev | Event Name | ***System-taxationOf-(User-paymentTo-User)*** | |
| | Event Nr | #59 | |
| | Explanation | This is a relational event that represents the system's action of taxing an inter-user sale of attention. | |
| | Subject Event | *User* | |
| | Object Event | *User* | |
| | Data Fields | *Taxed Amount* | The amount extracted by the system as tax. |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***Ransoming Event*** | | |
| | Event Nr | #60 | | |
| | Explanation | This event consists of the requesting by the CCP, to some peripheral peer, for it to periodically report back to CCP on specific user actions. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***Ransom Donation Event*** | 0 – ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | *User-insertionOf-MO* | 0 – ∞ |

| Proc Comp Ev | Event Name | ***Ransom Donation Event*** | | |
|---|---|---|---|---|
| | Event Nr | #61 | | |
| | Explanation | This event consists of a user donating some monetary resources to another user, on to pay for the ransom of some prospective MO, which is announced by some specific MO Ransom Announcement. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *User-likingOf-MORnsmAnn* | 1 |
| | | *Event-inclusionOf-Event* | *(User-likingOf-MORnsmAnn) -causationOf- (User-rewardingOf-User)* | 1 |
| | | *Event-inclusionOf-Event* | *User-rewardingOf-User* | 1 |
| | | *Event-inclusionOf-Event* | *System -taxatingOf- (User-rewardingOf-User)* | 1 |

| Rel Ev | Event Name | ***User-loadingOf-UAccount*** | |
|---|---|---|---|
| | Event Nr | #62 | |
| | Explanation | This is a relational event that represents a user action which consists of loading a user's account with monetary resources. | |
| | Subject Event | *User* | |
| | Object Event | *UAccount* | |
| | Data Fields | *LoadedAmount* | The amount loaded by the user. |

| Rel Ev | Event Name | ***User-unloadingOf-UAccount*** | |
|---|---|---|---|
| | Event Nr | #63 | |
| | Explanation | This is a relational event that represents a user action which consists of extracting monetary resources from his account. | |
| | Subject Event | *User* | |
| | Object Event | *UAccount* | |
| | Data Fields | *LoadedAmount* | The amount extracted by the user. |

| Rel Ev | Event Name | ***User-insertionOf-Peer*** | | |
|---|---|---|---|---|
| | Event Nr | #64 | | |
| | Explanation | This is a relational event that represents a user's initiative to insert a peer into the system. | | |
| | Subject Event | *User* | | |
| | Object Event | *Peer* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***Peer Registration Event*** | 0 - 1 |

| Rel Ev | Event Name | ***User-removalOf-Peer*** | | |
|---|---|---|---|---|
| | Event Nr | #65 | | |
| | Explanation | This is a relational event that represents a user's initiative to remove a peer from the system. | | |
| | Subject Event | *Peer* | | |
| | Object Event | *System* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***User Disconnection Event*** | 0 - ∞ |
| | | *Event-causationOf-Event* | ***MO Deregistration event*** | 0 - ∞ |
| | | *Event-causationOf-Event* | ***Peer Deregistration Event*** | 0 - 1 |

**D.1.2.2.1.2** *System Originated*

| Proc Comp Ev | Event Name | ***Report Requesting Event*** | | |
|---|---|---|---|---|
| | Event Nr | #66 | | |
| | Explanation | This is an internal system event. It consists of the requesting by the CCP, to some peripheral peer, for it to periodically report back to CCP on specific user actions. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *Peer-requestTo-Peer* | 1 |
| | | *Event-inclusionOf-Event* | *ERR -pertinenceTo- (Peer-requestTo-Peer)* | 1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***Reporting Event*** | 0 – ∞ |

| Proc Comp Ev | Event Name | ***Reporting Event*** | | |
|---|---|---|---|---|
| | Event Nr | #67 | | |
| | Explanation | This is an internal system event. It consists of a peripheral peer reporting back to the CCP (regarding some user action), in response to a previous Event Report Request received by the earlier from the latter. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *Peer-respondingTo-Peer* | 1 |
| | | *Event-inclusionOf-Event* | *ER -pertinenceTo- (Peer-respondingTo-Peer)* | 1 |

| Proc Comp Ev | Event Name | ***Users Evaluation Event*** | | |
|---|---|---|---|---|
| | Event Nr | #68 | | |
| | Explanation | This is a periodical, internal, system event which consists of an evaluation of all of the system users' conduit. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***User Evaluation Event*** | 1 – ∞ |

| Proc Comp Ev | | | | |
|---|---|---|---|---|
| | Event Name | ***User Evaluation Event*** | | |
| | Event Nr | #69 | | |
| | Explanation | This is an internal system event. It consists of the evaluation of a specific user's conduit, taking into consideration other users' misbehaviour reports regarding that user. It may lead to the expulsion, quarantining or dequarantining of the evaluated user. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *System-evaluationOf-User* | 1 |
| | | *Event-inclusionOf-Event* | *System-analysisOf-UserMissbehavReport* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *(System-analysisOf-UserMissbehavReport)-pertainmentTo-(System-evaluationOf-User)* | 0 – ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | *System-evictionOf-User* | 0 – 1 |
| | | *Event-initiationOf-Event* | *System-quarantiningOf-User* | 0 – 1 |
| | | *Event-terminationOf-Event* | *System-quarantiningOf-User* | 0 – 1 |

| Rel Ev | | | | |
|---|---|---|---|---|
| | Event Name | ***System-evictionOf-User*** | | |
| | Event Nr | #70 | | |
| | Explanation | This is an internal system event. It consists of the eviction of a user from the system's user community. This implies the removal of all his MOs and Peers. | | |
| | Subject Event | *System* | | |
| | Object Event | *User* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***User Deregistration Event*** | 0 – ∞ |

| Proc Comp Ev | | | | |
|---|---|---|---|---|
| | Event Name | ***MOs Evaluation Event*** | | |
| | Event Nr | #71 | | |
| | Explanation | This is a periodical, internal, system event which consists of an evaluation of all of the system MOs. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***MO Evaluation Event*** | 1 – ∞ |

| Proc Comp Ev | | | | |
|---|---|---|---|---|
| | Event Name | ***MO Evaluation Event*** | | |
| | Event Nr | #72 | | |
| | Explanation | This is an internal system event. It consists of the evaluation of a specific MO, taking into consideration users misbehaviour reports that concern it. It may lead to the eviction, quarantining or dequarantining or of the evaluated MO. | | |
| | Event's Relationships | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |

| | | | | |
|---|---|---|---|---|
| | as Subject | *Event-inclusionOf-Event* | *System-evaluationOf-MO* | 1 |
| | | *Event-inclusionOf-Event* | *System-analysisOf-UserMissbehavReport* | 0 – ∞ |
| | | *Event-inclusionOf-Event* | *(System-analysisOf-UserMissbehavReport)-pertainmentTo-(System-evaluationOf-MO)* | 0 – ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | *System-evictionOf-MO* | 0 – 1 |
| | | *Event-initiationOf-Event* | *System-quarantiningOf-MO* | 0 – 1 |
| | | *Event-terminationOf-Event* | *System-quarantiningOf-MO* | 0 – 1 |

| | | | | |
|---|---|---|---|---|
| Rel Ev | Event Name | ***System-evictionOf-MO*** | | |
| | Event Nr | #73 | | |
| | Explanation | This is an internally triggered relational system event. It represents the eviction of an MO from the system. | | |
| | Subject Event | *System* | | |
| | Object Event | *MO* | | |
| | Event's Relationships as Subject | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***MO Deregistration Process*** | 0 - 1 |

| | | | | |
|---|---|---|---|---|
| Proc Comp Ev | Event Name | ***UEL Servicing Session Event*** | | |
| | Event Nr | #74 | | |
| | Explanation | This event consists of the maintenance of UEL servicing responsabilities (over the peripheral peer collective) by a specific OCP. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *(Peer(OCP) -uelServicingOf-Peer(PP))* | 1 |
| | | *Event-inclusionOf-Event* | *Instruction OP IO -pertainmentTo-(Peer(OCP) -uelServicingOf-Peer(PP))* | 1 |

**D.1.2.2.2** *Registered Global UEL Entitary Complex Events*

| | | | |
|---|---|---|---|
| Ent Comp Ev | Event Name | ***User*** | |
| | Event Nr | #75 | |
| | Explanation | This event consists of a system human user. | |
| | Data Fields | User Type | The type of user (regular consumer/producer user or advertiser user) |
| | | Username | The user's username |
| | | Pub Key | The user's public key |
| | | Priv Key | The user's private key |

| Ent Comp Ev | Event Name | ***User Account*** | |
|---|---|---|---|
| | Event Nr | #76 | |
| | Explanation | This event consists of a the user account of a system user. | |
| | Data Fields | Account balance | The amount of momentary funds stored in the account |

| Ent Comp Ev | Event Name | ***Trusted System*** | | |
|---|---|---|---|---|
| | Event Nr | #77 | | |
| | Explanation | This event consists of another (P2PTube-like) system, which is trusted by the present system. | | |
| | Data Fields | *System Name* | The identifying name of the trusted system | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | **Peer (the CCP of the trusted system)** | 1 |

| Ent Comp Ev | Event Name | ***System*** | |
|---|---|---|---|
| | Event Nr | #78 | |
| | Explanation | This event consists of the actual system. | |
| | Data Fields | *System Name* | The identifying name of this system |

| Ent Comp Ev | Event Name | ***Operational IO*** | |
|---|---|---|---|
| | Event Nr | #79 | |
| | Explanation | This event consists of an operational information object in the system. | |
| | Data Fields | *OpIO Type* | The type of operational IO |
| | | *Tstamp* | Time stamp indication the validity termination date of the data contained in the operational IO |
| | | *Usig* | The signature of the user involved (if any) in the production of the operational IO |
| | | *Psig* | The signature of the peer involved (if any) in the production of the operational IO |
| | | *Ssig* | The system's (*CCP's* or *OCP's*) signature of the operational IO |

| Ent Comp Ev | Event Name | ***Search Query IO*** | |
|---|---|---|---|
| | Event Nr | #80 | |
| | Explanation | This event consists of a specific search query. Desides the data fields presented in the *Operational IO* event, its registry will also contain the one presented bellow. | |
| | Data Fields | *Search Query* | The search query string. |

| Ent Comp Ev | Event Name | ***MO*** | |
|---|---|---|---|
| | Event Nr | #81 | |
| | Explanation | This event consists of a media object. | |
| | Data Fields | *MO Usage Type* | The usage type of the media object (regular MO, advertisement MO, etc) |
| | | *Semantic Type* | The basic semantic category to which the MO pertains (comedy, drama, etc) |
| | | *Size* | The byte-wise size of the MO |
| | | *Duration* | The temporal length of the MO |
| | | *Usig* | The owner user's signature of the media object |
| | | *Ssig* | The system's (*CCP's*) signature of the MO |
| | | *Frag Nr* | The number of fragments the MO is broken into for redistribution |

| Ent Comp Ev | Event Name | ***MOFrag*** | |
|---|---|---|---|
| | Event Nr | #82 | |
| | Explanation | This event consists of a fragment of an MO. | |
| | Data Fields | *Size* | The byte-wise size of the MO fragment |
| | | *Ssig* | The system's (*CCP's*) signature of the MO fragment |
| | | *Frag Seq Nr* | The MO fragment's sequence number within the context of its "parent" MO |

| Ent Comp Ev | Event Name | ***MORansomAnnouncement*** | |
|---|---|---|---|
| | Event Nr | #83 | |
| | Explanation | This event consists of an announcement of the ransoming of a prospective future M. | |
| | Data Fields | *Semantic type* | The semantic type of the prospective MO |
| | | *Has Been Paid* | A boolean indication if the ransom has already been paid |
| | | *MO Delivered* | A boolean indication if the corresponding MO has already been delivered |

| Ent Comp Ev | Event Name | ***User Key Pair*** | |
|---|---|---|---|
| | Event Nr | #84 | |
| | Explanation | This event consists of a user's public and private key pair. | |
| | | *Pub Key* | The user's public key |
| | | *Priv Key* | The user's private key |

**D.1.2.3** *Registered Global Cross Layer Information*

| Proc Comp Ev | Event Name | ***Data Model Synchronization Event*** | | |
|---|---|---|---|---|
| | Event Nr | #85 | | |
| | Explanation | This is an internal system event. It consists of the updating, by the CCP, of the data structure copies located at all the relevant OCP, so that all data structure copies are coherent. | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *Peer-updatingOf-Peer* | 1 – ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-causationOf-Event* | ***Peer Evaluation Event*** | 0 – ∞ |

| Proc Comp Ev | Event Name | ***OCP Activity Reporting Event*** | | |
|---|---|---|---|---|
| | Event Nr | #86 | | |
| | Explanation | This is an internal system event. It consists of an OCP reporting to the CCP information regarding its workload | | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *Peer(OCP)-notificationTo-Peer(CCP)* | 1 |
| | | *Event-inclusionOf-Event* | *OCPActivityNotification -pertainmentTo- (Peer(OCP) -notificationTo- Peer(CCP))* | 1 – ∞ |

## **D.1.3***Registered Individual Peer Information*

The event registries, presented bellow, are temporarily stored at every peer to enable the coordination of their individual operation. They constitute the peer's individual model and are not stored at the core data structure.

**D.1.3.1** *Registered Individual PLL Information*

**D.1.3.1.1** *Registered Individual Relational and Procedural Complex Events*

| Proc Comp Ev | Event Name | ***Inter-Peer PLL Connection Event*** | | |
|---|---|---|---|---|
| | Event Nr | #87 | | |
| | Explanation | This event consists of the establishment of a PLL communicational connection between two peers. | | |
| | Parent Type | Peer PLL Operation Event | | |
| | | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *Peer-connectionTo-Peer* | 0 - 1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-initiationOf-Event* | ***Inter Peer PLL Comm Session Event*** | 0 – 1 |

| Proc Comp Ev | Event Name | ***Inter-Peer PLL Disconnection Event*** | | |
|---|---|---|---|---|
| | Event Nr | #88 | | |
| | Explanation | This event consists of the termination of a PLL communicational connection between two peers. | | |
| | Parent Type | Peer PLL Operation Event | | |
| | | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | *Peer-disconnectionFrom-Peer* | 0 - 1 |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-terminationOf-Event* | ***Inter Peer PLL Comm Session Event*** | 0 – 1 |

| Proc Comp Ev | Event Name | ***Inter Peer PLL Comm Session Event*** | | |
|---|---|---|---|---|
| | Event Nr | #89 | | |
| | Explanation | This event consists of a the maintenance of a communication session between two peers. | | |
| | Data Fields | Session Key | The secret key for the encryption of the messages exchanged during the communication session | |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-enablingOf-Event* | ***Inter Peer PLL Cooperation Event*** | 0 – ∞ |

| Proc Comp Ev | Event Name | ***Peer PLL Operation Event*** | | |
|---|---|---|---|---|
| | Event Nr | #90 | | |
| | Explanation | This event consists of the performing of some specific PLL operation by some peer. | | |
| | Data Fields | Op Type | The type of the operation | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***Inter Peer PLL Cooperation Event*** | 0 – ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-supportingOf-Event* | ***Peer UEL Operation Event*** | 0 – 1 |

| Proc Comp Ev | Event Name | ***Inter Peer PLL Cooperation Event*** | | |
|---|---|---|---|---|
| | Event Nr | #91 | | |
| | Explanation | This event consists of the cooperation between to peers (at the PLL level). | | |
| | Data Fields | Played Role | Client or server | |
| | | Coop Type | The type of the cooperation | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***PLL Msg Sending Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***PLL Msg Reception Event*** | 0 – ∞ |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-supportingOf-Event* | ***Inter Peer UEL Cooperation Event*** | 0 – ∞ |

| Proc Comp Ev | Event Name | ***PLL Msg Sending Event*** | | |
|---|---|---|---|---|
| | Event Nr | #92 | | |
| | Explanation | This event consists of the sending of a PLL message. | | |
| | Data Fields | Msg. Seq. Nr | The sequence number of the sent PLL message | |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-reactingTo-Event* | ***PLL Msg Reception Event*** | 0 - 1 |

| Proc Comp Ev | Event Name | ***PLL Msg Reception Event*** | | |
|---|---|---|---|---|
| | Event Nr | #9 | | |
| | Explanation | This event consists of the reception of a PLL message. | | |
| | Data Fields | Msg. Seq. Nr | The sequence number of the received PLL message | |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-reactingTo-Event* | ***PLL Msg Sending Event*** | 0 - 1 |

**D.1.3.2** *Registered Individual UEL Information*

**D.1.3.2.1** *Registered Individual Relational and Procedural Complex Events*

| Proc Comp Ev | Event Name | ***Peer UEL Operation Event*** | | |
|---|---|---|---|---|
| | Event Nr | #93 | | |
| | Explanation | This event consists of the performing of some specific UEL operation by some peer. | | |
| | Data Fields | Op Type | The type of the operation | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***Inter Peer UEL Cooperation Event*** | 0 – ∞ |

| Proc Comp Ev | Event Name | ***Inter Peer UEL Cooperation Event*** | | |
|---|---|---|---|---|
| | Event Nr | #94 | | |
| | Explanation | This event consists of the cooperation between to peers (at the UEL level). | | |
| | Data Fields | Plaid Role | Client or server | |
| | Event's Relationships as Subject | Structural | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-inclusionOf-Event* | ***UEL Msg Sending Event*** | 0 – ∞ |
| | | *Event-inclusionOf-Event* | ***UEL Msg Reception Event*** | 0 – ∞ |

| Proc Comp Ev | Event Name | ***UEL Msg Sending Event*** | | |
|---|---|---|---|---|
| | Event Nr | #95 | | |
| | Explanation | This event consists of the sending of a UEL message. | | |
| | Data Fields | Msg Seq Nr | The sequence number of the sent UEL message. | |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-reactingTo-Event* | ***UEL Msg Reception Event*** | 0 - 1 |

| Proc Comp Ev | Event Name | ***UEL Msg Reception Event*** | | |
|---|---|---|---|---|
| | Event Nr | #96 | | |
| | Explanation | This event consists of the reception of a UEL message. | | |
| | Data Fields | Msg Seq Nr | The sequence number of the received UEL message. | |
| | | Contextual | | |
| | | Connecting RelEv Name | Connected Object Event | Cardinality |
| | | *Event-reactingTo-Event* | ***UEL Msg Sending Event*** | 0 - 1 |

# References


[1] Techtarget, D. Wolff, "*peer-to-peer*", 2004, http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci212769,00.html.

[2] Stephanos Androutsellis-Theotokis, "*A Survey of Peer-to-Peer File Sharing Technologies*", Athens University of Economics and Business, 2002, http://www.spinellis.gr/pubs/jrnl/2004-ACMCS-p2p/html/AS04.html.

[3] IBM, T. Sundsted, "*The practice of peer-to-peer computing: Introduction and history*", 2001.

[4] OECD, "*OECD Information Technology Outlook – Peer To Peer Networks in OECD Countries – Pre-release of Section from Chapter 5 of the Information Technology Outlook*", 2004.

[5] OECD, "*OECD Information Technology Outlook*", 2006.

[6] Networks and Telecommunications Research Group at Trinity College Dublin, "*Website dedicated to P2P and networking technologies*", 2003, http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/Intro.html.

[7] Mp3newswire, R. Menta, "*RIAA Sues Music Startup Napster for $20 Billion*", 1999, http://www.mp3newswire.net/stories/napster.html.

[8] San Francisco Chronicle, E. Benny, "*Napster runs out of lives – judge rules against sale*", 2002.

[9] Gnutella Homepage, http://www.gnutella.com.

[10] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "*Freenet: A distributed anonymous information storage and retrieval system*", Proceedings of ICSI Workshop on Design Issues in Anonymity and Unobservability, 2000.

[11] Edonkey Network, http://www.edonkey2000.com (This site has been disabled for legal reasons).

[12] PIER Project Homepage, Berkeley, Computer Science Division, http://pier.cs.berkeley.edu/index.html.

[13] Edutella Project, http://www.edutella.org/edutella.shtml.

[14] Mac-P2P.com, "*Peer to Peer (P2P) Introduction and History*", 2003, http://www.mac-p2p.com/p2p-history.

[15] Gnutella Protocol Development Registration Site, Sourcefourge, http://rfc-gnutella.sourceforge.net/.

[16] "*The FastTrack Protocol*", http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack/PROTOCOL?rev=HEAD&content-type=text/vnd.viewcvs-markup, 2004.

[17] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, "*A Survey and Comparison of Peer-to-Peer Overlay Network Schemes*", Communications Surveys & Tutorials, IEEE, 2005.

[18] L. Garcés-Erice, E. W. Biersack, P. A. Felber, K. W. Ross, G. Urvoy-Keller, *"Hierarchical Peer-to-peer Systems*", Proceedings of ACM/IFIP Intl. Conference on Parallel and Distributed Computing, 2003.

[19] M. Schlosser, M. Sintek, S. Decker, W. Nejdl, "*HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks*", Proceedings of the First International Workshop on Agents and Peer-to-peer Computing (AP2PC), 2002.

[20] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "*SkipNet: a scalable overlay network with practical locality properties*", Proceedings of 4th USENIX Symposium on Internet Technologies and Systems (USITS), 2003.

[21] Q. Lv, S. Ratnasamy, and S. Shenker, "*Can heterogeneity make Gnutella scalable?*", In Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), MIT Faculty Club, 2002.

[22] Miguel Castro, Manuel Costa, and Antony Rowstron, "*Should we build Gnutella on a structured overlay?*", SIGCOMM Comput. Commun. Rev. 34, 1 (January 2004), 131-136, 2004.

[23] C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "*Search and replication in unstructured peer-to-peer networks*", Proceedings of the 16th international conference on Supercomputing, New York, 2001.

[24] B. Yang, H. Garcia-Molina, "*Improving search in peer-to-peer networks*", Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), 2002.

[25] P. Reynolds, A. Vahdat, "*Efficient peer-to-peer keyword searching*", Proceedings of IFIP/ACM Middleware, 2003.

[26] S. Rhea, C. Wells, and others, "*Maintenance-free Global Data Storage*", IEEE Internet Computing, Volume 5, pages 40 - 49, 2001.

[27] A. Ghodsi, "*Distributed k-ary System: Algorithms for Distributed Hash Tables*", Doctoral thesis, Chapter 6.1, KTH, Royal Institute of Technology, 2006.

[28] P. Druschel, A. Rowstron, "*Past: A large-scale, persistent peer-to-peer storage utility*", Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, 2001.

[29] M. Waldman, A. D. Rubin, L. F. Cranor, "*Publius: A robust, tamper-evident, censorship-resistant web publishing system*", Proceedings of the 9th USENIX Security Symposium, USA, 2000.

[30] S. Hand, T. Roscoe, "*Mnemosyne: Peer-to-peer steganographic storage*", Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), MIT Faculty Club, 2002.

[31] C. Peery, F. M. Cuenca-Acuna, R. P. Martin, T. D. Nguyen, "*Collaborative Management of Global Directories in P2P Systems*", Rutgers University, 2002.

[32] A. Muthitacharoen, R. Morris, T. M. Gil, B. Chen, "*Ivy: A Read/Write Peer-to-Peer File System*", 5th Symposium on Operating Systems Design and Implementation, USA, 2002.

[33] A. R. Butt, T. A. Johnson, Y. Zheng, Y. C. Hu "*Kosha: A Peer-to-Peer Enhancement for the Network File System*", Proceedings of the ACM/IEEE SC2004 Conference, 20004.

[34] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "*The Eigentrust algorithm for reputation management in p2p networks*", Proceedings of the Twelfth International World Wide Web Conference, 2003.

[35] M. Gupta, P. Judge, and M. Ammar, "*A reputation system for peer-to-peer networks*". NOSSDAV'03 Conference, California, 2003.

[36] L. Xiong and L. Liu, "*Building Trust in Decentralized Peer-to-Peer Electronic Communities*", 5th International Conference on Electronic Commerce Research, 2002.

[37] Liu, "*Free riding: A new challenge for peer-to-peer file sharing systems*", Proceedings of the 36th HICSS Conference, Hawaii, 2003.

[38] E. Damiani et al, "*A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks*", Proceedings of the 9th ACM conference on Computer and Communications Security (CCS'02), pages 207 to 216, 2002.

[39] B. Chun, Y. Fu, A. Vahdat, "*Bootstrapping a distributed computational economy with peer-to-peer bartering*", Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, 2003.

[40] B. Cohen, "*Incentives Build Robustness in BitTorrent*", Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems, California, 2003.

[41] K. Anagnostakis, M. Greenwald, "*Exchange-based incentive mechanisms for peer-to-peer file sharing*", 24th International Conference on Distributed Computing Systems (ICDCS'04), Tokyo, 2004.

[42] B. F. Cooper, H. Garcia-Molina, "*Peer-to-peer resource trading in a reliable distributed system*", Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), MIT Faculty Club, 2002.

[43] R. Dingledine, M. J. Freedman, D. Molnar, "*The FreeHaven project: Distributed anonymous storage service*", Proceeding of the Workshop on Design Issues in Anonymity and Unobservability, 2000.

[44] V. Vishnimurthy, S. Chandrakumar, E. Gun Sirer, "*Karma: A secure economic framework for p2p resource sharing*", Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, 2003.

[45] B. Yu, M. P. Singh, "*Incentive mechanisms for peer-to-peer systems*", Proceedings of the 2nd International Workshop on Agents and Peer-to-Peer Computing, 2003.

[46] B. Yang, H. Garcia-Molina, "*PPay: Micropayments for Peer to Peer Systems*", Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), 2003.

[47] O'Reilly, Andy Oram and others, "*Peer-to-Peer: Harnessing the Power of Disruptive Technologies*", 2001, ISBN 10: 0-596-00110-X | ISBN 13: 9780596001100.

[48] J. R. Douceur, "*The Sybil attack*", Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), MIT Faculty Club, 2002.

[49] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi, "*Threshold cryptography in P2P and MANETs: The case of access control*", Comput. Netw. 51, 12, pp 3632-3649, August 2007.

[50] S. Garfinkel, "*PGP: Pretty Good Privacy*", O'Reilly & Associates, Inc., Cambridge, MA,1995.

[51] V. Pathak, L. Iftode, "*Byzantine fault tolerant public key authentication in peer-to-peer systems*", Computer Networks. Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security,50(4):579–596, March 2006.

[52] J. Dinger, H. Hartenstein, "*Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration*", Proceedings of the First International Conference on Availability, Reliability and Security, 2006.

[53] R. Dingledine, M. J. Freedman, D. Molnar, "*Peer-to-Peer: Harnessing the Power of Disruptive Technologies*", O'Reilly, 2001.

[54] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, S. R. Gummadi, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao, "*Oceanstore: An architecture for global-scale persistent storage*", Proceedings of the Ninth international Conference on Architectural Support For Programming Languages and Operating Systems, 2000.

[55] W. J. Bolosky, J. R. Douceur, D. Ely, M. Theimer, "*Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs*", Proceedings of ACM SIGMETRICS 2000, 2000.

[56] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, "*Wide-Area Cooperative Storage with CFS*", Proceedings of the 18th ACM symposium on Operating systems principles, 2001.

[57] Shane Balfe, Amit D. Lakhani, and Kenneth G. Paterson, "*Trusted Computing: Providing Security for Peer-to-Peer Networks*", In Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P '05), IEEE Computer Society, Washington, DC, USA, 117-124.

[58] A. Datta, M.Hauswirth,K. Aberer, "*Beyond "web of trust": Enabling P2P E-Commerce*", Proceedings of IEEE International Conference on Electronic Commerce (CEC), 2003.

[59] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach, "*Secure routing for structured peer-to-peer overlay networks*", Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), 2002.

[60] T. Chothia, K. Chatzikokolakis, "*A Survey of Anonymous Peer-to-Peer File-Sharing*", Proceedings of the IFIP International Symposium on Network-Centric Ubiquitous Systems (NCUS 2005), Japan, 2005.

[61] M. Reiter, A. Rubin, "*Crowds: anonymity for web transactions*". ACM Transactions on Information and System Security (TISSEC), Volume 1, Issue 1, pages 66 - 92, 1998.

[62] D. Goldschlag, M. Reed, P. Syverson, "*Onion routing for anonymous and private Internet connections*", Communications of the ACM, 1999.

[63] D. Chaum, "*Untraceable electronic mail, return addresses and digital pseudonyms*", Communications of the ACM, Volume 24, Issue 2, pages: 84 - 90, 1981.

[64] A. Serjantov, "*Anonymizing censorship resistant systems*". Proceedings of the 1st International Workshop on Peer-to-Peer Systems, MIT Faculty Club, 2002.

[65] M. J. Freedman, E. Sit, J. Cates, R. Morris, "*Introducing tarzan, a peer-to-peer anonymizing network layer*", Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), MIT Faculty Club, 2002.

[66] S. Hazel, B. Wiley, "*Achord: A variant of the Chord lookup service for use in censorship resistant peer-to-peerpublishing systems*", Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), MIT Faculty Club, 2002.

[67] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica, "*Wide-area cooperative storage with CFS*", In Proc.ACM SOSP'01, Banff, Canada, October 2001.

[68] P. Maniatis, M. Baker, "*Secure History Preservation Through Timeline Entanglement*", Proc. 11th USENIX Security Symposium, SF, CA, USA (2002).

[69] Neil Daswani, Hector Garcia-Molina and Beverly Yang, "*Open Problems in Data-sharing Peer-to-peer Systems",* Technical Report, Stanford InfoLab, 9th International Conference on Database Theory (ICDT 2003) Siena, Italy, January 8-10, 2003.

[70] A. Shamir, "*How to share a secret*", Communications of the ACM, Volume 22, Issue 11, pages: 612 - 613, 1979.

[71] Slyck News, T. Mennecke, "*eDonkey2000 Nearly Double the Size of FastTrack*", 2005, http://www.slyck.com/news.php?story=814.

[72] Gnutella For Users Homepage, http://rakjar.de/gnufu/index.php/GnuFU_en.

[73] A. Jantunen, S. Peltotalo, J. Peltotalo, "*Peer-to-Peer Analysis State-of-the-art*", Tampere University of Technology, 2006, http://delco.cs.tut.fi/doc/other/p2p_analysis_v01.pdf.

[74] The BitTorrent Protocol Specification, http://www.bittorrent.org/beps/bep_0003.html.

[75] Azureus official Homepage, http://azureus.sourceforge.net.

[76] Azureus wiki page on DHT usage, http://azureuswiki.com/index.php/DHT.

[77] Kademlia specification at Sourceforge, http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html.

[78] I. Thomson, Vnunet.com , "*P2P network eDonkey to close*", 2005, http://www.vnunet.com/vnunet/news/2142972/p2p-company-falls.

[79] N. Mook, BetaNews, "*eDonkey Firm to Pay RIAA $30 Million*", 2006, http://www.betanews.com/article/eDonkey_Firm_to_Pay_RIAA_30_Million/1158093147.

[80] S. B. Handurukande, A.M. Kermarrec, F. Le Fessant, L. Massoulié and S. Patarin, "*Peer Sharing Behaviour in the eDonkey Network, and Implications for the Design of Server-less File Sharing Systems*", Proceedings of the 2006 EuroSys conference, 2006.

[81] Kazaa Media Desktop, http://www.kazaa.com.

[82] J. Liang, R. Kumar, K. W. Ross, "*The FastTrack overlay: A measurement study*", Computer Networks: The International Journal of Computer and Telecommunications Networking, 2006.

[83] Q. Liu, R. Safavi-Naini, N. P. Sheppard, "*Digital Rights Management for Content Distribution*", Proceedings of the Australasian information security workshop conference on ACSW frontiers, 2003.

[84] World Intellectual Property Organization, Standing Committee on Copyright and Related Rights, Tenth Session, J. P. Cunard, K. Hill, C. Barlas, "Cu*rre*nt *Developments in the Field of Digital Rights Management*", 2005.

[85] MEDIANET Project, "*DRM/CP Requirements of Selected Use Cases and Business Scenarios*", Deliverable DB.5.7.

[86] Open Digital Rights Language (ODRL) Initiative Homepage, http://odrl.net.

[87] Cover Pages, "*ODRL Version 1.0 Submitted to ISO/IEC MPEG for Rights Data Dictionary and Rights Expression Language (RDD-REL)*", http://xml.coverpages.org/ni2001-11-21-d.html.

[88] eXtensible Rights Markup Language Homepage, http://www.xrml.org.

[89] EURESCOM, Project OPERA, S. Wegner, "*Overview of the state-of-the-art at DRM Systems and Standardisation Activities*", 2002.

[90] R. G. González, "*A Semantic Web approach to Digital Rights Management*", PhD Thesis, Universitat Pompeu Fabra, Barcelona 2005.

[91] Creative Commons Homepage, http://creativecommons.org.

[92] OASIS XACML committee Homepage, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[93] Cover Pages Technology Reports, "*Extensible Access Control Markup Language*", http://xml.coverpages.org/xacml.html, 2005.

[94] Microsoft, "*Microsoft Next-Generation Secure Computing Base Technical FAQ*", http://www.microsoft.com/technet/archive/security/news/ngscb.mspx?mfr=true.

[95] Trusted Computing Group Homepage, https://www.trustedcomputinggroup.org/home.

[96] Windows Live ID Homepage, https://accountservices.passport.net/ppnetworkhome.srf.

[97] OpenID Homepage, http://openid.net.

[98] OpenID Foundation Homepage, http://openid.net/foundation.

[99] Shibboleth Homepage, http://shibboleth.internet2.edu.

[100] OASIS, "*OASIS Security Services (SAML)*", http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[101] E. Rodríguez, J. Delgado, "*Trust in event reporting mechanisms for DRM*", 3rd IEEE International Workshop on Digital Rights Management Impact on Consumer Communications (CCNC 2007), 2007.

[102] MPEG-21, "*Information Technology — Multimedia Framework — Part 15: Event Reporting*", ISO/IEC JTC 1/SC 29/WG 11, 2006.

[103] Open Mobile Alliance Homepage, http://www.openmobilealliance.org.

[104] CEN/ISSS (European Committee for Standardization / Information Society Standardization System), "*Digital Rights Management Final Report*", European Commission, 2003, http://ec.europa.eu/enterprise/ict/policy/doc/drm.pdf.

[105] Open Mobile Alliance, "*OMA-TS-DRM-DRM-V2_0-20050614-C*", 2005, http://www.openmobilealliance.org/technical/release_program/docs/DRM/V2_0-20050614-C/OMA-TS-DRM-DRM-V2_0-20050614-C.pdf.

[106] FindArticles.com, "*Vodafone Selects Industry Leading Digital Rights Management Solution from CoreMedia*", 2004, http://findarticles.com/p/articles/mi_m0EIN/is_2004_Nov_23/ai_n7073339.

[107] CoreMedia, "*VIVO, Brazil's largest mobile operator selects CoreMedia DRM*", http://www.coremedia.com/en/103250/vivo-brazils-largest-mobile-operator-selects-coremedia-drm.

[108] OPIMA Homepage, http://www.chiariglione.org/leonardo/standards/opima/index.htm.

[109] ISMA Homepage, http://www.isma.tv.

[110] ISMA, "*Internet Streaming Media Alliance Encryption and Authentication Version 1.1*", 2005, http://www.isma.tv/technology/TD00084.pdf.

[111] Michael LoBue, Reuters,"*MPEGIF and ISMA: ISMA Mission Accomplished, Residual Assets Assumed by MPEGIF*", 2010 , http://www.reuters.com/article/2010/04/27/idUS30029+27-Apr-2010+BW20100427

[112] MPEG, "*MPEG-4 IPMPX white paper ISO/IEC JTC 1/SC 29/WG 11*"

http://www.chiariglione.org/mpeg/technologies/mp04-ipx/index,htm.

[113] MPEG-21 Consortium, "*ISO/IEC FDIS 21000-5:2003(E) Information Technology — Multimedia Framework — Part 5: Rights Expression Language*", 2003.

[114] Digital Media Project Homepage, http://www.dmpf.org.

[115] The Digital Media Project, L. Chiariglione, Proposal for "*Approved Document No. 4, WD 1.1 – Technical Specification: Use Cases and Value Chains, ver. 3.0*", 2007.

[116] Microsoft, "*Architecture of Windows Media Rights Manager*", http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx.

[117] SCG AG Homepage, http://www.digicont.ch/c_index.html.

[118] SDC AG, "*Mobile Code Architecture and Digital Container Object*", http://www.digicont.ch/sdc_java_drm/core_technology/index.html.

[119] OpenIPMP Homepage, http://objectlab.com/clients/openipmp/index.htm.

[120] AXMEDIS project Homepage, http://www.axmedis.org.

[121] AXMEDIS project, "*AXMEDIS DRM FOR DUMMIES*", http://www.axmedis.org/documenti/view_documenti.php?doc_id=3964.

[122] Nate Anderson, "*Hacking Digital Rights Management*", Ars Technica, 2007, http://arstechnica.com/apple/news/2006/07/drmhacks.ars.

[123] Veoh Homepage, http://www.veoh.com.

[124] Jon Brodkin 2011, "*Appeals Court reaffirms DMCA protection for user-generated content*", Ars Technica, http://arstechnica.com/tech-policy/news/2011/12/appeals-court-reaffirms-dmca-protection-for-user-generated-content.ars.

[125] Babelgum Homepage, http://www.babelgum.com.

[126] Screendigest, "*Babel Networks to join increasingly crowded online video market*", http://www.screendigest.com/online_services/intelligence/broadband/updates/bi-070307-dcsj1/view.html.

[127] PaidContent Org, R. Andrews, "*Joost Rival Babelgum Opens Doors To P2P TV Viewers*", 2007, http://www.paidcontent.org/entry/419-joost-rival-babelgum-opens-doors-to-p2p-tv-viewers.

[128] News.com, G. Sandoval, "*Skype founders name new video start-up Joost*", 2007, http://www.news.com/Skype-founders-name-new-video-start-up-Joost/2100-1026_3-6150225.html.

[129] Staci D. Kramer, "*Joost Says It Has No Future As Portal, Enters White-Label Market; Volpi Out As CEO*", paidContent Org, 2009, http://paidcontent.org/article/419-joost-admits-no-future-as-portal-volpi-out-as-ceo-staff-cuts-white-labe.

[130] PPLive Homepage, http://www.pplive.com/en/index.html.

[131] ReelTime Homepage, http://www.reeltime.com.

[132] Livestation Homepage, http://www.livestation.com.

[133] NewTeeVee, J. Roettgers, "*Livestation Opens Up Beta Test, Focuses on News*", 2008, http://newteevee.com/2008/02/11/livestation-opens-up-beta-test-focuses-on-news.

[134] Livestation, "*Scalable High Quality Solution*", http://www.livestation.com/broadcast_platform?tracker=main_menu.

[135] TVNewSer, "*The Future of News Viewing*", http://www.mediabistro.com/tvnewser/nabrtnda_2008/the_future_of_news_viewing__82514.asp.

[136] InternetNews, S. M. Kerner, "*Imeem Launches P2P Social Network*", 2005, http://www.internetnews.com/xSP/article.php/3527381.

[137] Devin Leonard, "*Making free music pay off*", CNNMoney, 2008.http://money.cnn.com/2008/08/07/technology/imeem.fortune/index.htm.

[138] Eliot Van Buskirk, "*MySpace Music Acquires Shuttered Imeem Music Service*", Wired Magazine, 2009, http://www.wired.com/business/2009/12/myspace-music-acquires-shutters-imeem.

[139] BBC iPlayer Homepage, http://www.bbc.co.uk/iplayer.

[140] Kontiki Home Page, http://www.kontiki.com.

[141] The Register, C. Williams, "*BBC iPlayer for iPhone and iPod Touch is iGo*", 2008, http://www.theregister.co.uk/2008/03/07/iplayer_iphone_availablity.

[142] Channel Register, C. Williams, "*BBC iPlayer finally hits the streets*", http://www.channelregister.co.uk/2007/06/27/iplayer_launch, 2007.

[143] CNET News, D. Meyer, "*BBC iPlayer launch on schedule, despite DRM crack*", 2007, http://news.cnet.co.uk/software/0,39029694,49291676,00.htm.

[144] The New York Times, R. Levine, "*New Model for Sharing: Free Music With Ads*", 2007, http://www.nytimes.com/2007/04/23/technology/23qtrax.html.

[145] EMI Press Release, "*EMI Music becomes the first major music company to make its catalogue available to Qtrax: the world's first ad-supported, legitimate P2P service*", 2006, http://www.emigroup.com/Press/2006/press25.htm.

[146] Qtrax Home Page, http://music.qtrax.com.

[147] Eliot Van Buskirk, "Surprise*! Qtrax, The 'Free and Legal Music Downloads' Service, Is Back*", Wired Magazine, 2011, http://www.wired.com/epicenter/2011/03/qtrax-is-back.

[148] Sky Anytime Home Page, http://anytime.sky.com.

[149] Sky Home Page, http://www.sky.com.

[150] MPPGlobal Home Page, http://www.mppglobal.com/07_news/n_news60-BSkyB-and-MPP-Link-to-Provide-Multi-faceted-Web-Payments-for-Sky-Anytime.asp.

[151] ZDNet News, David Meyer, "*Sky hit by Windows Media DRM crack*", 2006, http://news.zdnet.com/2100-1009_22-149516.html.

[152] iMesh Home Page, http://www.imesh.com.

[153] BusinessWire, "*iMesh Inc. Subsidiary MusicLab Announces Beta Launch of BearShare Authorized Peer-to-Peer Service; Offering Includes ToGo Portable Music Subscription Service and Social Networking Features*", 2006,

http://www.businesswire.com/portal/site/google/index.jsp?ndmViewId=news_view&newsId=20060817005185&newsLang=en.

[154] Niall McKay, "*Peer-to-Peer Goes Legit*", Wired Magazine, 2005, http://www.wired.com/entertainment/music/news/2005/11/69457.

[155] TVU Networks Home Page, http://www.tvunetworks.com.

[156] StreamingMedia.com, "*TVU Showcases In-stream Video Ads for HP, Rolex and NetJets*", 2008, http://www.streamingmedia.com/press/view.asp?id=8623.

[157] Zattoo Home Page, http://zattoo.com.

[158] BroadbandTVNews, Julian Clover, "*Zattoo joins online TV crowd*", 2008, http://www.broadbandtvnews.com/?p=4191.

[159] PRNewsWire, "*Quick-start, Long-play Internet Television Arrives with Zattoo P2P IPTV*", 2006,

http://media.prnewswire.com/en/jsp/tradeshows/events.jsp?option=tradeshow&beat=BEAT_ALL&eventid=1001995&view=LATEST&resourceid=3218275.

[160] Constellation Protfolio Page, http://www.constellation.ch/eng/portfolio/constellation_i/zattoo.

[161] Jeremy Reimer, "*Windows Media DRM cracked*", Ars Technica, 2006, http://arstechnica.com/uncategorized/2006/08/7607.

[162] Bruce Schneier, "*Quickest Patch Ever*", Wired Magazine - Commentary, 2006, http://www.wired.com/politics/security/commentary/securitymatters/2006/09/71738.

[163] Anders Bylund, "*Apple's DRM cracked again*", Ars Technica, 2006, http://arstechnica.com/uncategorized/2006/08/7619.

[164] Recording Industry Association of America, "*Piracy: Online and On The Street*" 2008, http://www.riaa.com/physicalpiracy.php.

[165] K. Fisher, "*The Problem with MPAA's Shocking Piracy Numbers*", Ars Technica, 2006, http://arstechnica.com/old/content/2006/05/6761.ars.

[166] H. Castro, A. P. Alves, C. Serrão, B. Caraway, "*A New Paradigm for Content Producers*", IEEE Multimedia,vol.17, no.2, pp.90-93, April-June 2010.

[167] N. Anderson, "*Five Years of Failure: EFF Says RIAA Must Embrace New Model*", Ars Technica, 2008, http://arstechnica.com/tech-policy/news/2008/10/five-years-of-failure-eff-says-riaa-must-embrace-new-model.ars.

[168] E. Bangeman, "*P2P traffic shifts away from music, towards movies*", Ars Technica, 2007, http://arstechnica.com/tech-policy/news/2007/07/p2p-traffic-shifts-away-from-music-towards-movies.ars.

[169] T. Karagiannis et al., ''*Is P2P Dying or Just Hiding?*'', Proc. IEEE Globecom 2004—Global Internet and Next Generation Networks, vol. 3, 2004,pp. 1532-1538.

[170] R. Koenen et al., ''*The Long March to Interoperable Digital Rights Management*" Proc. IEEE,vol. 92, no. 6, 2004, pp. 883-897.

[171] M. Porter, "*Competitive Advantage—Creating and Sustaining Superior Performance*", Free Press, 2004.

[172] Z. Mutter, ''*Universal Offers DRM-free Music*'', PC Advisor, 2007, http://www.pcadvisor.co.uk/news/index.cfm?newsid=8675.

[173] EMI Group, ''*EMI Music Launches DRM-Free Superior Sound Quality Downloads Across Its Entire Digital Repertoire*'', 2007, http://www.emigroup.com/Press/2007/press18.htm.

[174] J. Golbeck, B. Parsia, and J.A. Hendler, "*Trust Networks on the Semantic Web*", In Proceedings of WWW (Posters), 2003.

[175] MPEG-21 Consortium, "*ISO/IEC FDIS 21000-17:2006(E) MPEG-21 - Part 17: Fragment Identification of MPEG Resources*", 2006.

[176] Carl Shapiro, Hal R. Varian, "*Information Rules: A Strategic Guide to the Network Economy*", Harvard Business Press, 1999.

[177] C. Anderson, "*The Long Tail: Why the Future of Business is Selling Less of More*", Hyperion, 2006.

[178] Helder Castro, A. Pimenta Alves, "*Support for Media Content Production and Distribution in the Internet Era*", First Workshop on Interdisciplinary Research in New Media, 2007.

[179] H. Castro, M. T. Andrade, A. P. Alves, "*Governed Media Distribution based on Nonrestrictive DRM*", International Conference on Telecommunications and Multimedia, Ierapetra, Crete, Greece, 2008.

[180] Helder Castro, Artur P. Alves, "*A P2P Content Delivery System for Alternative Business Models*", NWeSP 2011, Spain, 2011.

[181] H. Castro, A. P. Alves and M. T. Andrade, "*Reliable P2P Content Delivery for Alternative Business Models*", International Journal of Computer Information Systems and Industrial Management Applications, vol. 5, pp. 11–29, 2013 (accepted).

[182] H. Castro, A. P. Alves, "*Cognitive Object Format*", International Conference on Knowledge Engineering and Ontology Development, Funchal, Madeira, Portugal, 2009.

[183] Helder Castro, Maria Teresa Andrade, Fernando Almeida, Giuseppe Tropea, Nicola Blefari Melazzi,Leonardo Chiariglione, Aziz S. Mousas and Dimitra I. Kaklaman, "*Exploring Semantic Relationships Across Internet Resources*", NWeSP 2011, Spain, 2011.

[184] Teresa Andrade, Helder Castro, Edoardo Radica, Giuseppe Tropea, "*Preliminary input contribution for DID extension*",ISO/IEC JTC1/SC29/WG11, March 2011.

[185] Giuseppe Tropea, Helder Castro, "*Amendment for DII Digital Item Semantic Relationships*", ISO/IEC JTC1/SC29/WG11, November 2011.

[186] Teresa Andrade, Helder Castro, Leonardo Chiariglione, Aziz Moussas, Giuseppe Tropea, "*Proposal to extend MPEG-21 DII with means to support a semantically explicit declaration of relationships between Digital Items*", ISO/IEC JTC1/SC29/WG11, July 2011.

[187] Teresa Andrade, Helder Castro, Leonardo Chiariglione, Aziz Moussas, Giuseppe Tropea, "*Information technology — Multimedia framework (MPEG-21) — Part 3: Digital Item Identification, AMENDMENT 2: Digital Item Semantic Relationships*", ISO/IEC JTC1/SC29/WG11, January 2012.

[188] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "*A scalable content addressable network*", Processings of the ACM SIGCOMM, 2001.

[189] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "*Chord: A scalable peer-to-peer lookup protocol for internet applications*", Proceedings of the ACM SIGCOMM, California, 2001.

[190] C. Plaxton, R. Rajaraman, A. Richa, "*Accessing nearby copies of replicated objects in a distributed environment*" in Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures,1997.

[191] P. Maymounkov, D. Maziéres, "*Kademlia: A peer-to-peer information system based on the xor metric*", Proceedings of IPTPS02, USA, 2002.

[192] H. Castro, M.T. Andrade, F. Almeida, G. Troppea, N.B. Melazzi, A.S. Mousas, and D.I. Kaklamani, "*Semantically Connected Web Resources with MPEG-21*", Springer Multimedia Tools and Applications (submitted).