FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Video Broadcast Using Secure IP Multicast Techniques

António Alberto dos Santos Pinto

Dissertação submetida para satisfação parcial dos requisitos do grau de mestre em

Redes e Serviços de Comunicação

Dissertação realizada sob a supervisão do

Professor Doutor Manuel Alberto Pereira Ricardo,

do Departamento de Engenharia Electrotécnica e de Computadores

da Faculdade de Engenharia da Universidade do Porto

Porto, Julho de 2005

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Video Broadcast Using Secure IP Multicast Techniques

António Alberto dos Santos Pinto

Licenciado em Engenharia Informática

pelo Instituto Superior de Engenharia do Porto

Dissertação submetida para satisfação parcial dos requisitos do grau de mestre em

Redes e Serviços de Comunicação

Dissertação realizada sob a supervisão do

Professor Doutor Manuel Alberto Pereira Ricardo,

do Departamento de Engenharia Electrotécnica e de Computadores

da Faculdade de Engenharia da Universidade do Porto

Porto, Julho de 2005

To Carla, Maria João and Pedro Daniel

Resumo

A evolução tecnológica tem conduzido as telecomunicações para cenários de redes completamente baseadas em IP, em que os serviços são todos transportados sob a forma de pacotes IP. Entre estes encontra-se a difusão de vídeo que se espera que alcance os seus utilizadores com níveis de qualidade e segurança superior aos disponibilizados pelos actuais serviços analógicos de televisão por cabo. Existem já arquitecturas para a transmissão de conteúdos multimédia sobre IP, que incluem protocolos como o *Real-time Transfer Protocol* (RTP). Uma aproximação possível para transmitir múltiplos canais de vídeo poderá ser a utilização de grupos *multicast*, em que cada cliente indique explicitamente se pretende receber um canal de vídeo.

O IP *multicast* seguro, em definição pelo grupo de trabalho MSEC do IETF, pode ser uma solução para este problema. Aqui, apenas os utilizadores portadores da chave criptográfica correcta podem aceder e descodificar um canal de vídeo. O IP *multicast* seguro usa criptografia simétrica, com chave partilhada por todos os membros do grupo; obriga ainda que a chave do grupo seja renovada sempre que a composição deste muda. Para poder ser adoptado na difusão de vídeo, o IP *multicast* seguro necessita de se adaptar a cenários de transmissão de múltiplos canais de vídeo, em que alguns clientes apenas estão autorizados a visionar alguns dos canais transmitidos. Estes novos cenários implicam que (1) o controlo de acessos é feito por canal de vídeo, e que (2) a rede transporte canais de vídeo que sejam visionados.

Estes cenários foram usados para definir o trabalho desenvolvido nesta tese de mestrado e conduziram a 4 resultados principais: (1) uma arquitectura de um sistema de transmissão de vídeo, (2) um elemento de rede capaz de filtrar tráfego *multicast*, (3) um mecanismo de gestão de grupos e de controlo de acessos e (4) um protótipo do sistema de vídeo. O primeiro resultado consiste num sistema de transmissão de vídeo capaz de distribuir múltiplos canais de vídeo sobre IP *multicast* seguro, utilizando cifra simétrica mas mantendo o controlo de acessos independente para cada canal de vídeo. O segundo resultado é um elemento de rede capaz de filtrar transmissões *multicast* de vídeos com base nos pedidos dos clientes; não transmite canais de vídeo não solicitados na rede de acesso. O terceiro resultado permite que um cliente entre num grupo *multicast*, obtendo uma chave individual que lhe permite decifrar o canal de vídeo. O quarto resultado consiste num ambiente de teste composto por 3 *PCs*, cada um simulando um componente do sistema de transmissão de vídeo e implementando protocolos para transmissão em *multicast* seguro, gestão de grupos e controlo de acessos. O segundo resultado é, do nosso ponto de vista, novo e constitui uma contribuição original.

Abstract

The technological evolution is leading telecommunications to all-IP network scenarios, where multiple services are transported as IP packets. Among these services is the broadcast of video which is expected to reach users with quality and security better than those available today in analog cable TV. Architectures for transmitting multimedia contents over IP networks have already been defined by IETF, which include protocols such as the RTP. A possible mechanism for broadcast multiple video channels over IP could be to use IP multicast, and let each client to decide about the reception of a video channel.

The secure IP multicast being specified by IETF MSEC working group is a candidate solution for this problem. Using it, users can access and decode a video channel only if they have a correct cryptographic key. The secure IP multicast solution uses symmetric encryption techniques, which lead to a single key for all the group members; it also requires group key renewal when the group composition changes. In order to be usable in video broadcast (e.g. cable TV), secure IP multicast needs also to address scenarios supporting multiple video channel transmission, where some clients are authorized to view only a subset of the video channels transmitted. These new scenarios imply that (1) the user access is controlled per video channel, and (2) the network needs not to transport channels which are not visioned.

The work carried out in this Master thesis addressed these problems and lead us to 4 main results: (1) a video transmission system architecture, (2) a multicast filtering network element, (3) a group management and access control mechanism, and (4) a video system prototype. The first result consists of a video transmission system capable of delivering multiple video channels over secure IP multicast, using symmetric encryption but still allowing the access control of clients per video channel. The second is a network element that can filter the multicast video transmission based on the client characteristics and requests; it filters the transmission of un-requested video channels into the last mile network. The third enables a client to join a multicast group; by manifesting its interest, the client receives an individual cryptographic key which gives him access to the ciphered video transmission. The fourth result consists in a 3 PC testbed, each PC emulating a video system component and implementing protocols for secure multicast transmission, group management and access control. We claim that, to the best of our knowledge, the second result is new, and it constitutes an original contribution.

Acknowledgements

I would like to start by thanking my supervisor, Prof. Manuel Ricardo, for his help, guidance, availability and patience, especially when considering the work spent in reviewing this thesis and the short time window to do so.

Concluding, I would like to thank my family for their support and understanding. To my wife, Carla, for all her efforts in creating conditions that allowed me to work, which was not always appreciated.

Contents

List of figures

List of tables

Glossary

3GPP	3rd Generation Project Partnership
AAA	Authentication, Authorization and Accounting
ADSL	Asymmetric Digital Subscriber Line
BM-SC	Broadcast Multicast Service Center
CEK	Channel Encryption Key
CS	Cipher Sequences
DEP	Dual-Encryption Protocol
DVB	Digital Video Broadcast
DVRMP	Distance Vector Multicast Routing Protocol
FDM	Frequency Division Multiplexing
GCKS	Group Controller / Key Server
GDOI	Group Domain of Interpretation
GGSN	Gateway GPRS Support Network
GSA	Group Security Association
GSAKMP	Group Secure Association Key Management Protocol
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IGKMP	Intra-Domain Group Key Management Protocol
IGMP	Internet Group Management Protocol
IMGP	Implicitly Managed Group Protocol
IP	Internet Protocol
IPSEC	Internet Protocol Security
IPv4	Internet Protocol, version 4
IPv6	Internet Protocol, version 6
KDP	Key Distribution Protocol
LAN	Local Area Network

MARKS	Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences
MBMS	Multimedia Broadcast/Multicast Services
MD	Multicast Deflector
MDE	Multicast Deflector Engine
MIKEY	Multimedia Internet Keying
MLD	Multicast Listener Discovery Protocol
MOSPF	Multicast Open Shortest Path First
MSC	Message Sequence Charts
MSEC	Multicast Security
OSI	Open System Interconnection
OTP	One-time Pad
PIM	Protocol Independent Multicast
PIM-DM	Protocol Independent Multicast - Dense Mode
PIM-SM	Protocol Independent Multicast - Sparse Mode
POP	Post Office Protocol
PRNG	Pseudorandom Number Generators
QoS	Quality of Service
RP	Rendezvous Point
RPF	Reverse Path Forwarding
RTCP	Real-time Control Protocol
RTP	Real-time Transfer Protocol
RTSP	Real-time Streaming Protocol
SA	Security Association
SEK	Session Encryption Key
SIP	Session Initiation Protocol
SMKD	Scalable Multicast Key Distribution
SMTP	Simple Mail Transfer Protocol
SPI	Security Parameter Index

SRTP	Secure Real-time Transfer Protocol
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TESLA	Timed Efficient Stream Loss-tolerant Authentication Protocol
TLS	Transport Layer Security
TTL	Time-to-live
UDP	User Datagram Protocol
UE	User Equipment
VoIP	Voice over IP
WAN	Wide Area Network

1. Introduction

The technological evolution is leading telecommunications to all-IP network scenarios, where multiple services are transported as IP packets. Solutions for transmitting, video, voice and multimedia contents in IP packets already exist, and the Real-time Transfer Protocol (RTP) [1] is a key component of these solutions. In some networks, not over-provisioned, there might be a need for traffic prioritization in order to provide Quality of Service (QoS) to the real-time services.

Among these services is the broadcast of video which is expected to reach users with quality and security better than those provided today in analog cable TV. The latter is mainly based on coaxial cable and enables the transmission of several video channels, independently of these channels being visualized. The simultaneous channel transmission is accomplished by modulating the analog signal of each channel at a different frequency, and using Frequency Division Multiplexing (FDM) techniques. The all-channel, all-time transmission leads to bandwidth waste, but it enables short response times when clients switch between channels.

Considering an all-IP network, a possible way of transmitting multiple video channels would be to use multicast groups, and to enable each client to decide if he wants to receive a video channel. This solution consists in one-to-many transmission, where each receiver shows its interest in receiving the data by joining a multicast group, which can be represented by an IP multicast address. Each receiver must know previously the multicast address of the channel and, by issuing a multicast group join request to its gateway router, it forces the modification of the multicast routing tables which instruct the transit routers to route data towards the receiver.

Security is a major concern of the new generation of IP networks. Key issues are authentication and encryption. The first allows access control and its main objective is to guarantee the identity of each element involved in communications; the second allows confidentiality by transforming plain data into unintelligible data, through the use of computer algorithms. Together, they enable the elimination of problems such as eavesdropping, unauthorized access and data integrity violation. There are security solutions at multiple communication layers (Data link, IP, Transport and Application). The 802.1X protocol, for instance, provides authentication at the data link layer, by not allowing unauthorized users to access the network resources. Internet Protocol Security (IPSec) provides data confidentiality at network level. Secure Socket Layer (SSL) or Transport Layer Security (TLS) enable confidentiality and authentication at the transport layer and provide secure communications to application protocols that do not implement security mechanism by themselves; Hypertext Transfer Protocol (HTTP), Post Office Protocol (POP), or Simple Mail Transfer Protocol (SMTP) are examples of these protocols. However, each application can support security by using its own mechanisms.

Traditional analog cable TV provides also security, mainly at the physical level, by using private cables or key locking distribution boxes. Despite this effort, it is easy to make a cable derivation and go undetected for long periods of time. More elaborated security mechanisms are those used in pay-per-view channels, where video signal unscrambling devices are used. This extra security is compromised by the need to transmit the channel to all the destinations, even if the costumer does not subscribe the channel. In this case, illegal equipments can be used to de-scramble the video channel.

Secure IP multicast [2] may be a solution for this problem. In this solution, only the users having the correct cryptographic key can access and decode a video channel. The standard secure IP multicast solution uses symmetric encryption techniques, which lead to a single cryptographic key for all the group members; besides, it requires group key renewal when the group composition changes. The secure multicast arises as a solution, but additional aspects must be considered in scenarios of multiple video channel transmission, where some clients are authorized to view only a subset of the channels transmitted. These aspects are twofold: 1) the access control needs to be implemented per video channel basis; 2) the network resources consumed by multiple video transmissions, some of which not visioned, need to be reduced. The first aspect can be solved by adopting a secure multicast group management protocol that provides each user with a set of cryptographic keys, one for each subscribed channel, that are unique and not shared with the other members of the group. The second aspect can be solved based on the idea that if a channel is not viewed, it needs not to be transmitted; this solution

can be implemented by filtering the channel at some network element in the edge of the network, freeing the bandwidth that becomes available for other services.

1.1. Reference Scenario

The reference scenario adopted in this thesis is presented in Figure 1. It consists of set of video servers, a video distribution system, and a set of clients. Each server generates a video channel; the video is distributed as a stream of bytes, whose format is non-specified but must be transported by IP packets. The video distribution system retransmits a received video to the last mile network only if there are authorized clients interested in receiving it; the video distribution system must also be capable of processing video requests sent by clients, authenticate clients, and implement access control mechanisms; besides, it distributes video decryption keys to authorized clients. The client interacts with the video distribution system in order to authenticate itself, request videos, receive video as IP packets, and display them.

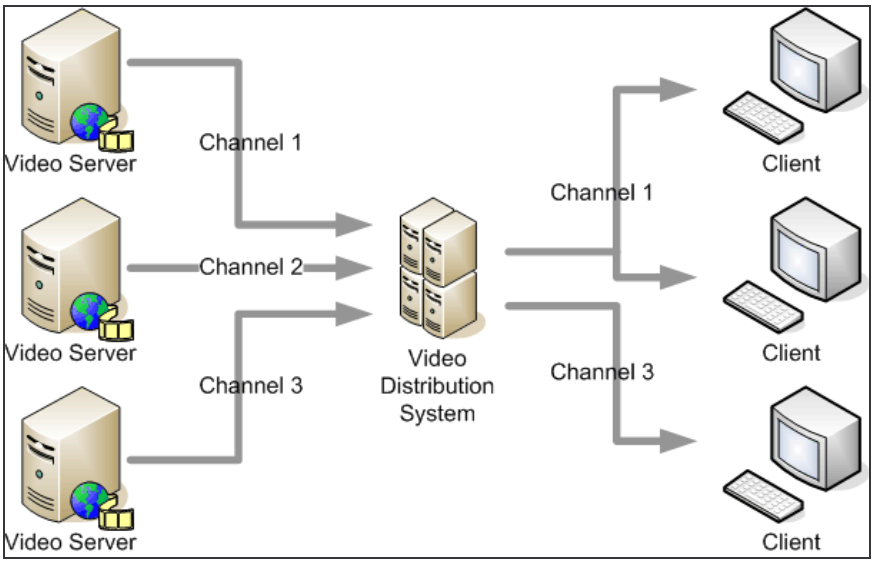

Figure 1 – Reference scenario.

From the security point of view some assumptions are made. At the video server there is no need for security besides that required for any Internet connected computer (e.g. firewall), because it resides in the service provider network, assumed to be reasonably secure. The stream sent by the video server is encrypted using a key pre-shared with the video distribution system; it implies that the video distribution system assume as authentic a video server holding the correct pre-shared key. The stream sent by the video distribution system to the last mile network is encrypted with a new key, different from the original pre-shared key. The client will assume as authentic any message signed with the public key of its serving video distribution system.

1.2. Objectives

The main objectives of this thesis are to:

1. Define a system capable of generating and distributing multiple video channels over a secure IP multicast infrastructure. The access of individual users to the video channels must be granted in a per channel basis.

2. Optimize of network usage over the *last mile* network, so only the channels required by the clients are transmitted.

1.3. Results and Contributions

The work carried out in this Master thesis lead us to 4 main results:

1. **Video transmission system architecture**. This architecture consists of a video transmission system capable of delivering several video channels over secure IP multicast, using symmetric encryption but still enabling the access control of clients per video channel.

2. **Multicast filtering network element**. A network element which can filter multicast video transmission based on the client characteristics and requests. It does not transmit un-requested video channels into the last mile network.

3. **Group management and access control mechanism**. This mechanism enables a client to join a multicast group. By manifesting its interest, the client receives an individual cryptographic key which gives him access to the coded video transmission.

4. **Video system prototype**. It consists in a 3 PC testbed, each PC emulating a relevant video system component and implementing protocols for secure multicast transmission, group management and access control. One PC acts as a Video Server, another as the Video Distributing System, and the third emulates a Client, as shown in Figure 1.

We claim that, to the best of our knowledge, the result number 2, multicast filtering network element, is new and it may constitute an original contribution.

1.4. Organization of the thesis

The following 3 chapters describe the state-of-the-art of the technologies relevant to the work presented in this thesis; they include multicast transmission in Chapter 2, video transmission

over IP networks in Chapter 3, and security technologies in Chapter 4. The Video Distribution System, that is the object of this thesis, is presented in Chapter 5, starting by the presentation of the system requirements, followed by the system specification and design and then by its implementation. The results of the functional and performance tests, obtained in our testbed, are presented and discussed in the Chapter 6. Concluding this thesis, Chapter 7 makes global considerations, characterizes the results and discusses future work.

2. Multicast

Internet Protocol multicast is a one-to-many communication mode which enables the optimized usage of network resources without adding extra work to the source. The key concept of a multicast transmission is the multicast group, which represents the recipients of a multicast data flow. An host can be a member of a multicast group and receive the multicast data, by sending the join message of the Internet Group Management Protocol (IGMP) [3], as shown in Figure 2. The source of a multicast group transmits its packets to the destination multicast address; the source is the sender's unicast address.

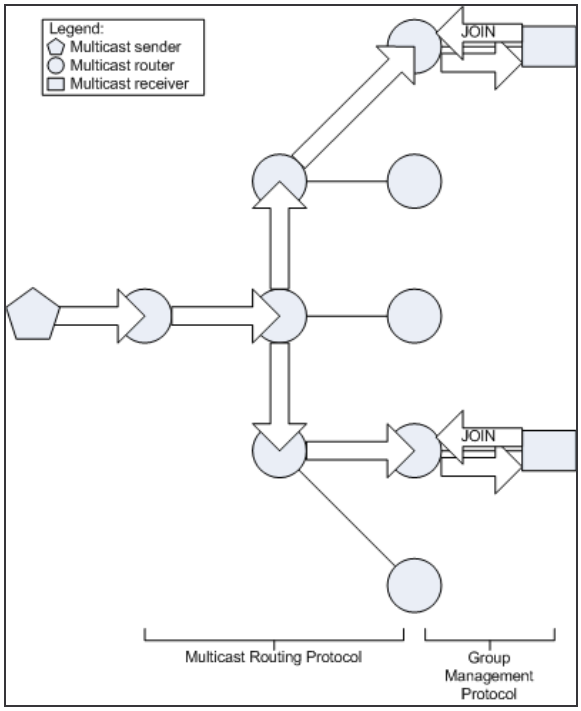

Figure 2 – Multicast transmission

Multicast groups are represented by IP multicast addresses in the range 224.0.0.0 to 239.255.255.255 (IPv4, Class D). The entity responsible for assigning these addresses, the Internet Assigned Numbers Authority (IANA), has reserved some of them, in the range 224.0.0.0 to 224.0.0.255, to routing protocols and other protocols in the local network. Some examples of these protocols are given in Table 1. Address spaces for global and limited scopes are also defined by IANA. The addresses ranging from 224.0.1.0 to 238.255.255.255 are global multicast; they are used for regular multicast transmissions over the Internet. The addresses ranging from 239.0.0.0 to 239.255.255.255 are used in multicast communications with limited reach; in this case, the router filters this traffic according to a user specified domain and stops it from being transmitted to other Autonomous Systems.

Address	Reserved for
224.0.0.1	All Systems on this Subnet
224.0.0.2	All Routers on this Subnet
224.0.0.5	OSPF Routers
224.0.0.9	RIP2 Routers
224.0.0.10	IGRP Routers

Table 1 – Reserved local multicast address

2.1. Multicast Group Management Protocols

Group communications demand group creation and group management. Two main protocols address these issues: the IGMP, and the Multicast Listener Discovery Protocol (MLD) [4, 5]. IGMP is the multicast group management protocol for IP version 4 (IPv4) networks, and MLD addresses IP version 6 (IPv6) networks.

2.1.1. Internet Group Management Protocol

In its version 1, IGMP defines that a host wanting to belong to a multicast group must send a message to the address 224.0.0.1 (“All systems on this subnet”), informing the local multicast router about its interest. Besides handling this request, this router polls periodically the local network to check if the host maintains its interest; if not, the router stops transmitting the multicast group data and informs neighbor routers about it. Two types of IGMP messages are used for these purposes:

- Membership report, used by hosts to indicate their intention to join a group;
- Membership query, which is sent periodically by routers in order to verify if there is at least one host interested in receiving the multicast data flow.

In IGMP version 1 group, the departures from the multicast group are based on timeout and induce some latency; in order to reduce it, IGMP version 2 introduces a group leave message, which informs the local multicast router about the intention of a member to leave the group. Besides the new leave message, the IGMP version 2 works basically as version 1 and, to be retro-compatible, a new version of the Membership report was also introduced. Therefore, two new types of messages were defined in version 2:

- Membership report, used by hosts to indicate their interest in joining a group;
- Leave group, used by hosts to indicate their intention to leave the group.

The latest version of the IGMP protocol, version 3, resulted in a profound adaptation of the protocol. It introduces the possibility of a host indicating the list of sources from which it wants to receive multicast traffic. A new type of message was then introduced in version 3:

- Membership report, used by a host to indicate its interest in joining a group and specify the source(s) from which it wants to receive data.

2.1.2. Multicast Listener Discovery Protocol

Similarly to IGMP on IPv4 networks, MLD enables IPv6 routers to discover the presence of multicast listeners on directly attached links, and to discover the multicast addresses of interest to those listeners. MLDv2 is a translation of IGMPv3 to IPv6 semantics. Multicast traffic filtering by source address is enabled by MLDv2 in some formats; using it, a multicast listener can describe both the sources it wants to receive multicast traffic from, as well as the sources that it wants to exclude. MLD, version 1, supports 3 message types:

- Multicast Listener Query, sent periodically by routers in order to verify that there is at least one host interested in receiving the traffic of a multicast group;
- Multicast Listener Report, used by hosts to indicate their interest in receiving the traffic of a multicast group;
- Multicast Listener Done, used by hosts to indicate the termination of its interest in receiving traffic from a multicast group.

MLD, version 2, became part of the Internet Control Message Protocol for IPv6 (ICMPv6) [6]. MLDv2 messages have as source address a link-local IPv6 address with a hop limit of 1 and an Hop-by-Hop Options header with the Router Alert option set. MLDv2 introduces a new message type:

- Version 2 Multicast Listener Report, used by hosts to indicate their interest in the traffic of a multicast group and to specify traffic filtering options.

2.2. Multicast Routing

The ability to create and manage groups of communicating nodes exists in several scenarios. In IP networks, for instance, an IP address mask may be used to define a group of IP devices with some geographical or organizational affinity. Multicast groups are widely spread, may be global or localized, and their size is a priori unknown. In order to enable this type of communications, routing mechanisms are required. Multicast routing protocols rely on protocols such as the multicast group management protocols described above and on IP routing protocols such as [7, 8]. Some well known routing protocols are: the Distance Vector Multicast Routing Protocol (DVMRP) [9], the Multicast Open Shortest Path First (MOSPF) [10, 11], and the Protocol Independent Multicast (PIM). The latter can be deployed in two modes, the dense mode (PIM-DM) [12], and the sparse mode (PIM-SM) [13].

The protocol independency claimed by PIM's name, refers to IP routing protocols; any unicast routing protocol capable of building unicast routing tables can be used by PIM. Unlike other multicast routing protocols, PIM does not build a routing table for multicast traffic, but it relies on unicast routing tables, and uses the Reverse Path Forwarding Check (RPF Check) technique to determine if the multicast traffic should be forwarded or not. The RPF Check consists in comparing the multicast packet source address with its arriving link. If there is a match, then the packet is forwarded to the destination link according to its destination address and the unicast routing table; if there is no match, the packet is dropped. Unlike other multicast routing protocols such as DVMRP, PIM does not send or receive multicast routing updates.

2.2.1. Protocol Independent Multicast - Dense Mode

The dense operational mode of the PIM protocol in a router forwards multicast traffic through its connected links so that the traffic reaches all the network elements. The downstream routers having no multicast listeners must prune unwanted traffic. PIM-DM assumes that there are multicast listeners in all downstream links and forwards through them. If a prune message is received from a neighbor downstream router, the multicast traffic is stopped in the link through which the prune message arrived. Multicast routing state information is thus based on the prune messages received by routers.

2.2.2. Protocol Independent Multicast - Sparse Mode

The PIM-SM works in opposition to the PIM-DM, by not assuming listeners in every network element. It makes listeners to explicitly request multicast data in order to start sending it. PIM-SM uses a concept named Rendezvous Point (RP), which represents a central element in the multicast group communication scenario. All sender and listeners must register there in order to be able to send or receive multicast traffic. RP reside on routers. PIM-SM is particularly adequate to scenarios where listeners are scattered over geographically distant networks, since PIM-SM sends multicast data only to the network regions (listeners) requesting the multicast traffic.

2.3. Layer 2 Multicast

An Ethernet switch normally treats multicast as broadcast traffic; it retransmits the multicast packets through all the ports, and does not take advantage of existing layer 2 multicast facilities. Nevertheless a set of layer two multicast facilities are available, which include the IANA multicast MAC address range, the IGMP snooping and some commercial solutions such as Cisco's Group Management Protocol.

IGMP snooping consists in listening IGMP traffic by the switch and by analyzing IGMP messages exchanged between hosts and routers. When an host Join Request is captured, the switch updates its multicast table, by associating the multicast address to the requesting port number. When the switch captures an IGMP leave message, it removes the respective entry from the multicast table. The multicast table is also updated according to IGMP query messages sent by routers. The drawback of IGMP snooping is that it requires analysis at network layer. Considering that IGMP messages are multicast, the switch must process all multicast traffic, making IGMP snooping an high processing task.

A network card usually receives frames whose destination address equals the card MAC address[1] or frames having the eighth bit of the destination address set to one[2], as shown in Figure 3. In order to ease layer 2 multicast, IANA also reserved an address space for multicast traffic, ranging from 01-00-53-00-00-00 to 01-00-5E-7F-FF-FF. Considering that, the least significant 23 bits are used to map the 23 least significant bits of IP multicast address. An

---

[1] MAC address – Medium access control layer address, also known has physical address.

[2] This bit in the MAC address is also known as the broadcast/multicast bit and indicates a multicast or broadcast frame. These types of frames are processed by all the network interface cards.

address mapping example is in Figure 3, where the IP address 239.0.0.1 is mapped into the 01-00-5E-00-00-01 MAC address.

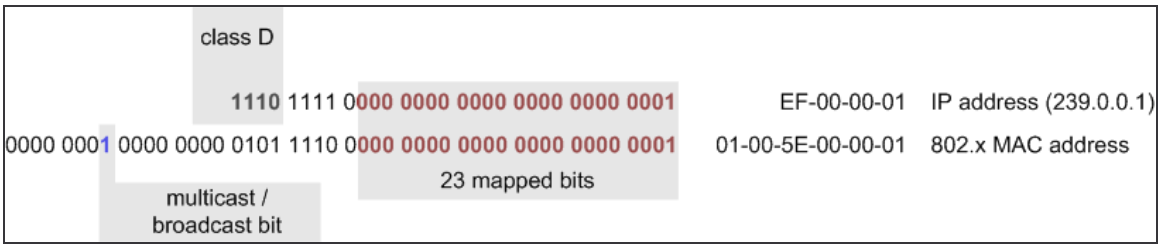

Figure 3 – IP multicast / 802.x MAC address mapping example

Despite of the benefits of this mapping technique, the loss of precision originated by mapping only 23 out of 32 bits of the IP addresses, implies that a MAC multicast address may represent multiple IP multicast addresses. As shown in Table 2, this ambiguity may originate additional processing time, since the decision of discarding or not forwarding a packet cannot be taken at the layer 2.

IP multicast / MAC address mapping	
224.0.0.1	
224.128.0.1	
225.0.0.1	
225.128.0.1	01-00-5E-00-00-01
…	
239.0.0.1	
239.128.0.1	

Table 2 – Address mapping ambiguity

2.4. Multicast in 3rd Generation Networks

The mobile device evolution combined with increasing radio bandwidths is making multimedia content distribution welcome by users and service providers. With this interest in mind, the 3rd Generation Project Partnership (3GPP) started to develop a standard to enable multicast multimedia services on the 3rd generation of mobile networks. The standard is the Multimedia Broadcast/Multicast Services (MBMS) [14] and it is currently under development.

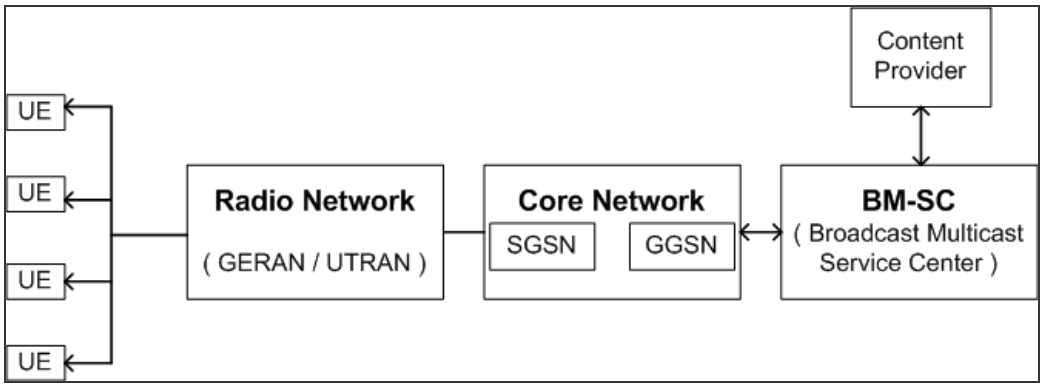

Figure 4 – MBMS Service architecture

The main objective of the MBMS is to standardize components and interfaces in its architecture, in order to promote synergies and enable interoperations between network operators, content providers and terminal and network equipments manufacturers [15]. The architecture proposed by 3GPP for MBMS services is shown in Figure 4 and consisting in the addition of a new functional entity called Broadcast Multicast Service Center (BM-SC) and in the adaptation of other components in order to cope with the MBMS standard. The BM-SC is the entity responsible for multicast service provisioning and delivery, and provides an entry point to multimedia content providers.

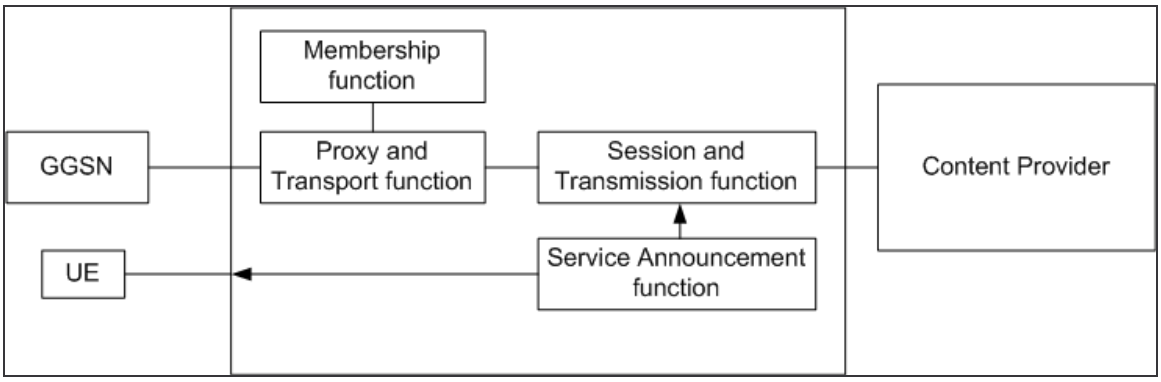

Figure 5 – BM-SC functional structure

The BM-SC represented in Figure 5 is composed of four main functional groups: 1) Membership function; 2) Session and Transmission function; 3) Proxy and Transport function; and 4) Service Announcement function. The Membership function authorizes the user equipments (UE) that request an MBMS service and, possibly, collects subscription data of MBMS service users, and generates charging records of MBMS service usage. The Session and Transmission function triggers transport bearers MBMS sessions, provides MBMS transmission or retransmissions, provides Gateway GPRS Support Network (GGSN) with multicast transport level parameters (e.g.: QoS parameters), authenticates external content providers and receives data from them. The Proxy and Transport function resides in a signaling proxy agent interoperating between the Membership function and the Session and

Transport Function; it has the capability of routing between multiple physical networks so that the different signaling interactions are transparent to the GGSN. The purpose of the Service Announcement function is to notify users about multicast and broadcast MBMS service availability, and to enable UE to access media and session descriptions.

Content deliveries via both unicast and MBMS bearers are depicted in Figure 6. In a typical content delivery through unicast bearers there are the same number of flows as there are users receiving content (Figure 6.a), thus limiting the maximum number of active users to the resources available. Considerable network resources can be saved with the adoption of an MBMS bearer (Figure 6.b), where there is only one flow between the content server and the GGSN, one tunnel per SGSN, and one radio bearer per set of UEs sharing the same radio resources, thus enabling a larger number of simultaneous users.

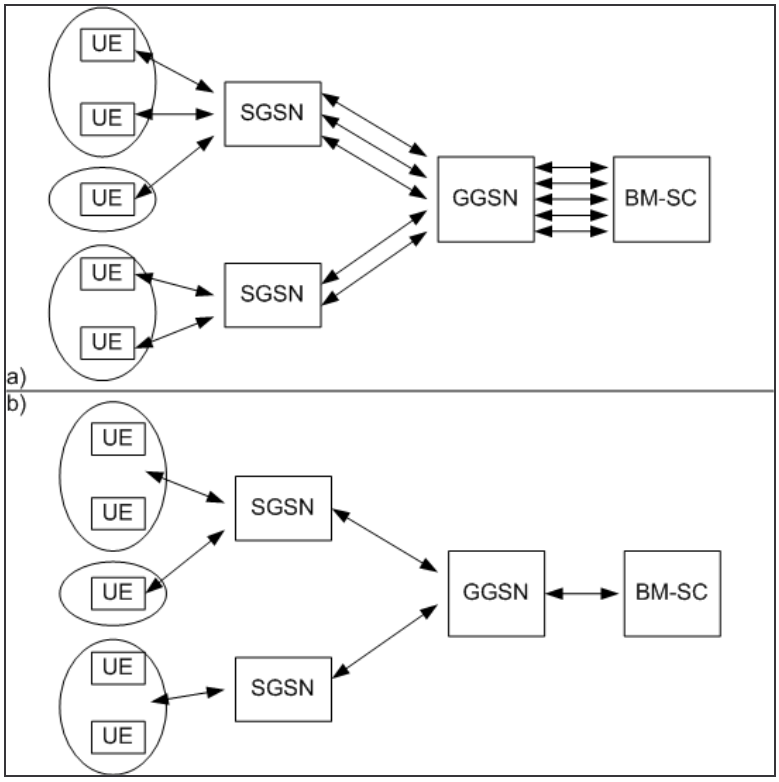

Figure 6 – Unicast/MBMS bearer content delivery

2.5. Multicast in Digital Video Broadcast Networks

Satellite communication technologies, where Digital Video Broadcast (DVB) [16] is included, are unidirectional broadcast networks adequate to services such as TV and radio broadcasting, for which there is no need for an uplink or back channel. Multicast traffic depends heavily in bidirectional communication, since the group management protocols consist in message exchanges between multicast receivers and gateway routers, and multicast routing protocols rely on the exchange of routing information between routers. The support of IP multicast over

satellite communications may imply modifications to standard multicast protocols, namely to the IGMP multicast group management protocol and to multicast routing protocols [17].

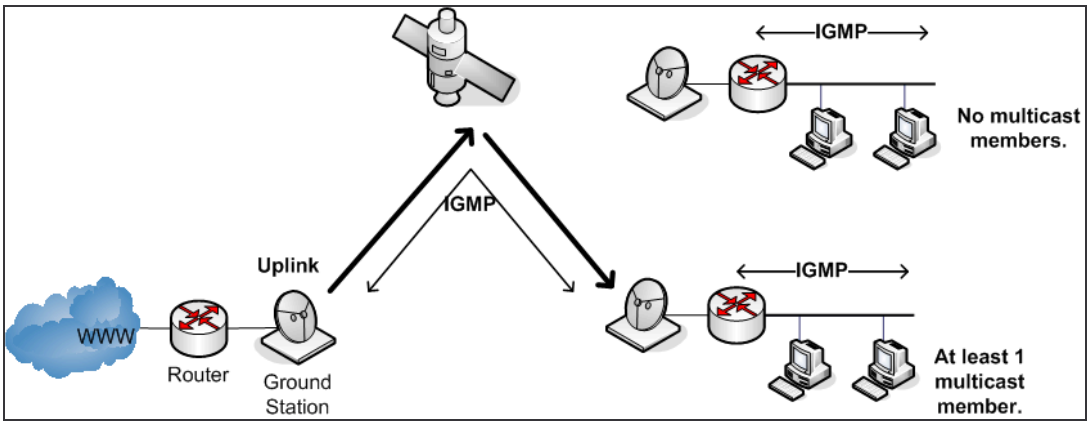

Figure 7 – IGMP over satellite

Multicast group management protocols, such as IGMP, work well in Local Area Networks (LANs) because all multicast listeners receive IGMP packets, avoiding the flooding of the network with multiple copies of IGMP Report packets. This does not happen in satellite communications, as one ground station cannot listen to the other stations and it is impracticable to duplicate IGMP Reports because it would mean a significant waste of network resources, namely where a large number of receivers is considered. A simple solution consists in the static configuration of the forwarding of multicast traffic through the satellite link to each downlink router, clearing the air interface of IGMP traffic (Figure 7). An alternative solution consists in transmitting over the air only selected IGMP messages; if an IGMP Report message is received by an uplink router after the reception of an IGMP Query message, then the IGMP Report is transmitted over the satellite link to all ground stations. IGMP snooping can be used in scenarios where there is no router in the downlink side.

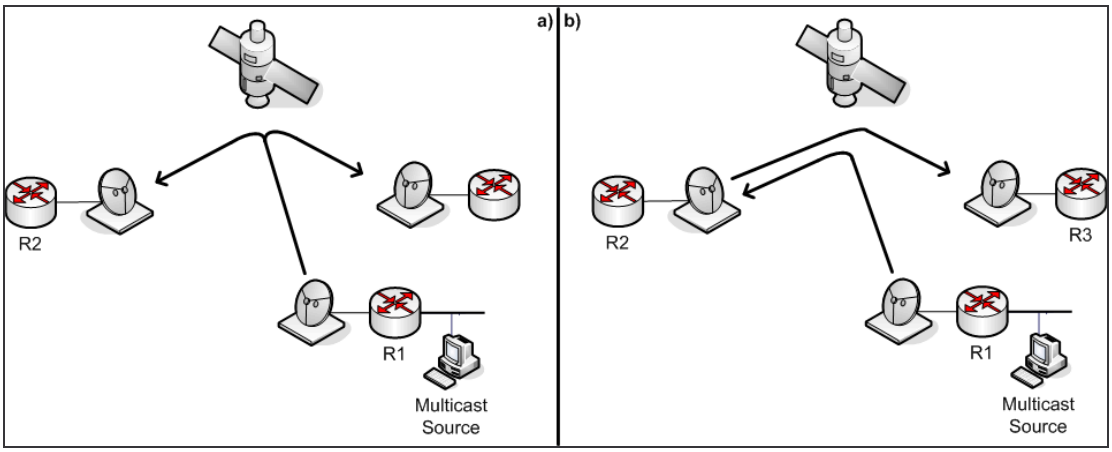

Figure 8 – Multicast routing through satellite

Standard multicast routing protocols such as PIM-DM and DVMRP work by initially flooding the network and, after that, start pruning links where there are no interested receivers. This

type of flood and prune algorithm may not work well in satellite environments, particularly if the air interface is not IGMP aware. The scenario represented in Figure 8 clearly demonstrates it, as the router R1 initially floods the network (Figure 8.a), and then starts pruning till it only sends to the router R2 which, in turn, send the multicast data back through the satellite to router R3 (Figure 8.b) making the multicast data go through the air interface twice and wasting network resources.

3. Video transmission over IP

The technological evolution is leading telecommunications to all-IP network scenarios, where multiple services are transported as IP packets. Solutions for transmitting video, voice, and multimedia contents over IP packets already exist. RTP and Voice over IP (VoIP) are examples of a mechanism for transmitting real-time data and a solution for conveying voice. IP network behavior is maintained since the network continues to be a packet network. There might be, however, a need for providing QoS to these services so that they can be perceived correctly by end users.

For video over IP, the control of network parameters such as bandwidth, packet delay and packet loss may be required; this control can be difficult to provide if we consider a large number of video consumers. The multicast transmission of video over IP seems to be an highly expected service; it addresses video, and enables bandwidth optimization by sending only once for all the group receivers. It also has some drawbacks since it hardens individual receiver quality control, which could be assured by unicast transmission.

3.1. Video streaming

Video streaming, as defined in [18], consists of a video "being played out while parts of the contents are received and decoded" and, thus, avoiding a full video download before decoding and visualization. Video streaming quality is significantly affected by the link quality and load; these aspects influence the bandwidth available, the delay variation and, possibly, the packet loss. Efforts have been made to support video streaming over IP. These efforts and solutions include network, transport, and session protocols.

3.2. Protocols for video streaming

The protocols presented here are not used exclusively for video streaming over IP; they can be used by other real-time applications over IP. They represent the Internet Engineering Task Force (IETF) multimedia architecture and include signaling protocols (SIP [19], RTSP [20]) and transport protocols (RTP, RTCP). RTP is used to transport data, i.e., the media stream; the remaining protocols are used to control the behavior of data transmission. Despite being used to control RTP sessions, RTCP can also transport media streams.

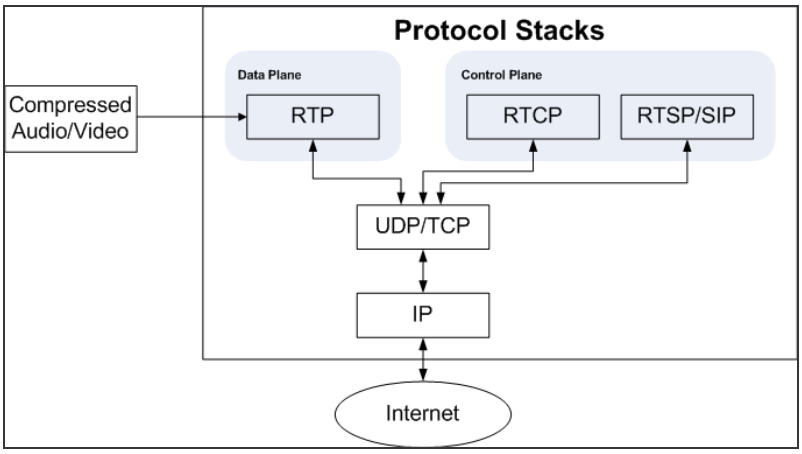

Figure 9 – Protocol stacks for media streaming

3.2.1. Real-time Transport Protocol (RTP)

The RTP protocol is an end-to-end transport protocol suited for real-time applications (e.g. video streaming), that can be used in multicast or unicast. It does not guarantee QoS or reserve resources by itself; for that purpose, it relies on lower protocols. RTP provides time-stamping, sequence numbering, payload type identification, and source identification. It is usually accompanied by the Real-time Control Protocol (RTCP), which monitors data packet delivery and enables operations such as participant identification, QoS feedback, control packet scaling, inter-media synchronization, and minimal session control information.

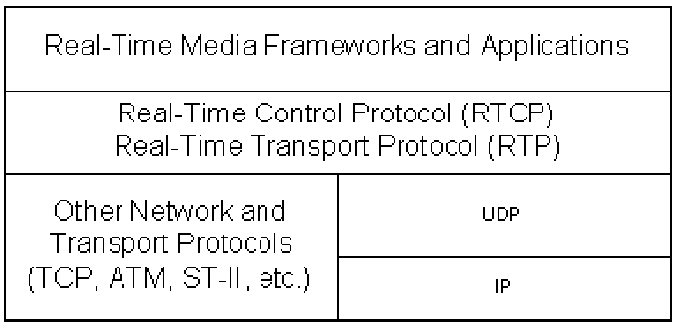

Figure 10 – RTP architecture

RTP is mainly used over UDP, in order to take advantage of data multiplexing and multicast transmission. Nevertheless, RTP can be used over other transport or network protocols as shown in Figure 10. The multi-user conference was the main scenario addressed during the development of RTP protocol, but RTP can have other applications benefiting from its features, such as continuous data storage for network backup systems, interactive and/or distributed simulations, and continuous control or measurement applications.

RTP adopts the concept of session, where each session carries a single data flow. Multiple sessions are required when multiple flows need to be transmitted. Some of these streams may need to be synchronized, has it happens with audio and video in a video telephony. Stream synchronization is made at the application layer based on the RTP time-stamping functionality. Because UDP does not sequence packets, RTP must order them at the receiver; for that purpose, it uses the sequence number of packets, marked by the sender. The sequence numbering functionality also enables packet loss detection. Each RTP packet identifies its payload type and its source using fields in the RTP packet header. The payload type identification eases the decoding process. The source identification enables the receivers to rapid distinguish among different sources.

RTCP, the RTP companion protocol, enables session participants to provide feedback to senders on the session reception quality by sending messages periodically. Messages sent by receivers are named receiver reports, while the messages sent by sources are named sender reports. These reports contain information such as RTP packets lost since the last report, total number of packets lost, jitter in packet arrival, and delay since last sender's report. This information enables the estimation of transmission rates, localization of congestion points, and the evaluation of network performance [18]. RTCP can also provide a more human-friendly mechanism for source identification by providing textual descriptions. In order to prevent bottlenecks and to scale, the number of packets sent from the control plane is based on a percentage of the total session bandwidth (5%), ¼ of which for sender reports and ¾ for receiver reports. Synchronization between associated media streams are accomplished based on information of both time and relative timestamps.

RTP was developed mainly to provide support for multimedia conferences, and its main types of actors are placed at the ends, the receivers and senders, known as *end systems*. Middle elements were also predicted, in order to support new functionalities; these are the *mixers* and the *translators*. A *translator* is aimed at changing the stream coding; the *mixer* aims at combining a set of streams into a single stream.

3.2.2. Real Time Streaming Protocol

The main function of the RTSP protocol is to provide VCR[3]-like control of video streaming sessions, but it also enables the selection of the stream transport mode (UDP, TCP, unicast, multicast), and supports stream establishment and control between servers and clients. RSTP is mainly used for stream control of multiple simultaneous media streams, either live or recorded. Its main function are those of a "remote control": stop, pause, play, fast forward and fast backward; provide functionalities such as session information retrieval, and the notification of clients and servers about the availability of new media contents which may be aggregated to established sessions.

3.2.3. Session Initiation Protocol (SIP)

SIP is used to start and terminate sessions between one or more participants. The key difference between SIP and RTSP is that SIP supports user mobility through user requests proxying and redirections to the user's current position. SIP is a textual client-server protocol where the clients issue requests and servers elaborate responses; both message types use HTTP [21] like syntax. Analogous to HTTP, a SIP request represents a form of *method* invocation on the server; there are six methods defined in [19], being the most relevant the INVITE method, which is used by clients to start a session with a server.

The SIP architecture consists of two main component types, user agents and network servers. User agents can act either as clients or servers of the protocol; the user agent client is used to start a session, and the user agent server is used to receive a session. Both user agents support peer-to-peer operation. The network servers can be of three types: proxy, redirect or registrar. A SIP proxy server behaves in the same way as a HTTP proxy server, i.e., it forwards the user agent requests to the next server. A SIP redirect server, after receiving the user agent request, finds the next server and informs the user agent, instructing it to redirect its request to the next server. A SIP registrar server enables user agent location registration and it processes user agents REGISTER requests.

---

[3] VCR – Video Cassette Recorder

4. Security

The concepts discussed in this chapter are related to network security and data protection, while in transit in a set of interconnected networks. These concepts include cryptographic techniques such as symmetric and asymmetric encryption. The security aspects of group communications scenarios with individual access control to the data sent to the group is also covered.

4.1. Cryptography

Symmetric encryption was the only type available prior to 1970[4]. It is based on key sharing between a sender and a receiver, and on an encryption algorithm that uses the shared key to transform intelligible data into unintelligible data, and vice-versa. The performance of this encryption scheme depends strongly on the encryption algorithm and on the length of the key. In turn, these requirements imply the use of a secure channel to distribute the key, and assume that the encryption/decryption algorithm is known by both parts [22].

Two major drawbacks unfold from the use of symmetric cryptography: 1) key disclosure to third parties compromises all communications; 2) both parties are equal (symmetric), and the receivers are able to forge traffic claiming it came from the sender. Public-key cryptography arises as a different approach and it uses cryptographic algorithms that require sets of two

[4] Public-key cryptography was invented around the year 1970.

keys: an encryption/signature verification key (public key); and a decryption/signature creation key (private key).

Public-key cryptography might suggest better security or appear as a replacement of symmetric encryption, but this is not true. The security of an encryption scheme depends essentially in the size of the key used; on the other hand, symmetric encryption is much faster than public-key encryption. For these reasons, public-key cryptography is used as a complement to symmetric cryptography.

4.1.1. Symmetric encryption

Symmetric encryption is traditionally used for message confidentiality in communications between workstations in a LAN environment, communications between LANs, and communications through wide area network (WAN). In these scenarios two major alternatives unfold: link encryption, and end-to-end encryption. Link encryption occurs at the layers 1 or 2 of the OSI model, and it is link dependent. End-to-end encryption is made at layers 3, 4, 6 or 7 of the OSI model and occurs between original source and final destination. Link encryption can protect and dissimulate traffic flows; end-to-end encryption cannot since header must be preserved in order to the data reach its destination. End-to-end encryption protects data over the entire path. So, in order to protect and authenticate data over the entire path, but maintaining the traffic patterns and characteristics inaccessible, both techniques may need to be used.

Key distribution is central to symmetric encryption because the source and the destination, and only them, must share a secret cryptographic key. Several key distribution schemes exist. A source can, for instance, generate a key and deliver it to the destination; a third party may also be used for that purpose. An older key may also be used to exchange the newly generated key. Other issues related to secure key distribution are automatic key distribution, decentralized key distribution, and usage of distributed keys. Automatic key distribution services can speed up and even protect key distribution, but require that the users trust the system. Decentralized key distribution can make the key distribution service more scalable but may imply a hierarchy of key distributors that must interoperate and trust each other. As soon as a key is delivered to the interested party, the control and the protection of that key is no longer under the responsibility of the key distribution service.

Cryptography makes intensive use of random numbers, which are used for generating keys, nonces[5], key streams for one-time pad[6], and others. Random generation of security numbers are usually uniformly distributed and need to be unpredictable; that is, the number generation must be regular, and the future generated number sequences must not be inferred from pastly generated numbers. A good solution is to base random number generation in the natural randomness of the real world by monitoring continuous random events such as radio noise, audio noise, or radiation counters. Other solutions are based on software such as Pseudorandom Number Generators (PRNG), which consists in creating nearby random numbers; in fact, these numbers are not truly random, but can pass *randomness* testing.

4.1.2. Public-key encryption

Public-key cryptography involves an encryption algorithm using two different keys. One of them is known only by the recipient (private-key), and the other is public (public-key). The public-key can be used either to encrypt messages or to verify digital signatures. The private-key can be used either to decrypt messages or to create digital signatures. The security parties are not symmetrical because those in charge of encrypting messages or verifying digital signatures cannot decrypt messages or create digital signatures. The reasons that lead to the development of public-key cryptography are twofold: key distribution, and digital signatures. In this case, key distribution can be made by untrusted third parties, and digital signatures may be easily confirmed.

The public-key algorithms working with two keys comply usually with the following characteristics:

- The discovery of the decryption key (private-key), knowing the algorithm and the encryption key (public-key), must be computationally unfeasible;
- The message encryption, when the encryption key is known, must be fast;
- The message decryption, when the decryption key is known, must be fast;
- Either key may be used to encrypt messages, being the other used to decrypt messages.

---

[5] A nonce is a randomly generated number used in some authentication protocols to prevent that an ill intentioned person captures and resends packets (replay) to impersonate others.

[6] One-time pad (OTP) is an encryption method that uses a key with the same size as the text to encrypt.

Public-key encryption algorithms take advantage of number theory problems which are easy to solve in one way but very difficult in the other way. For instance, the use of exponentiations and logarithms, or the use of multiplications and factoring. Typically the hard ways to solve the problem are publicly known, but they are just made too difficult to be used in practice. Attacks based on the testing of all possible keys (brute force attacks) are always possible. Thus, the protection relies on using large keys that will lead to large computation times.

4.2. Secure multicast

A secure multicast architecture is strongly influenced by the group size. The architecture defined in [23], for instance, is considered efficient in communications of small ad-hoc groups. The MSEC[7] architecture defined in [24] is being developed for large groups.

4.2.1. Multicast Security Group

The MSEC is an active IETF working group, which addresses secure group communications. This group considers three main problems: data confidentiality, group management and security, and group policy management and enforcement [25]. The MSEC solution consists of a set of blocks, which one addressing a problem. Secure group applications, can build solutions by combining these blocks.

The data confidentiality block addresses data security transforms and provides authentication and confidentiality to the group communications. While confidentiality is easily obtained through the use of symmetric encryption, data authentication requires more complex approaches since data encryption key needs to be shared among all the group members. Public-key encryption solves this problem but it is slower than symmetric encryption; hence, it may not be adequate to real-time traffic. The Timed Efficient Stream Loss-tolerant Authentication Protocol (TESLA) [26] is the authentication block proposed by MSEC for that purpose. It maintains the benefits of symmetric encryption, but it is unable to cope with immediate authentication. Senders, in TESLA, adopt a time relaxed authentication by using a chain of keys to sign data and by sending the signing key after the time interval it was used to sign the data. The first key is the unique key that is signed digitally.

---

[7] MSEC – IETF active working group on multicast security. MSEC working group is available online at the following URL: http://www.ietf.org/html.charters/msec-charter.html

In order to encrypt, decrypt, sign or authenticate messages, the group elements need to previously agree in a set of parameters such as encryption key and algorithms; this negotiation is carried out under the responsibility of the group management and security block. The central element in group management and key distribution is the Group Controller / Key Server (GCKS) described in [24], which provides key and data encryption keys to new members, after their authentication. The MSEC is currently developing three of these key management blocks: the Group Domain of Interpretation (GDOI) [27] that enables the creation and the management of Security Associations (SAs) for IPSec and other network or application layer protocols, the Multimedia Internet Keying (MIKEY) [28] that addresses the specifics of real-time multimedia applications and can be tunneled over SIP [19], and the Group Secure Association Key Management Protocol (GSAKMP) [29] that enable group policy specification and dissemination.

The third block addresses group policy management and enforcement, and defines policies based on the cryptographic key information holders, encryption algorithms, and on the authorization of the policy creator. This information is gathered into a policy token that, totally or partially, is sent to all group members using the GSAKMP protocol defined in [30].

4.2.1.1. Multicast Security Group Architecture

MSEC designs its secure multicast architecture as an end-to-end architecture, consisting of senders and receivers. The architecture is independent of multicast routing protocols such as [13], and of multicast admission control protocols such as [3] and [31]. This architecture, shown in Figure 11, is composed of four elements: Policy server, GCKS, Sender, and Receiver. They are distributed among three functional areas: Multicast security policies, Group key management, and Multicast data handling.

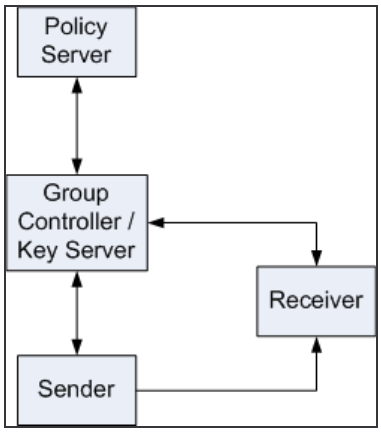

Figure 11 - Centralized Multicast Security Reference Framework

The GCKS element is the central security element in the MSEC reference framework. Its main responsibilities include cryptographic keys generation, key distribution and

management, and user authentication and authorization of the members wanting to be part of a secure multicast group. It also validates and enforces the security policies defined for the secure multicast group.

The Sender element is the multicast data source element, and it sends data to a secure multicast group. Depending on the multicast group type and scenario, there can be only one authorized sender (1-to-N multicast groups), several authorized senders (M-to-N multicast groups), or all members can be authorized senders. Interaction with the GCKS element is mandatory for authentication, authorization and cryptographic key distribution services access. The GCKS services used by the Receiver are basically the same used by the Sender, but the Receiver also needs to obtain keying material; for that end, it must first authenticate and be authorized.

The Policy Server element stores and defines multicast group security policies, and it is responsible for propagating policies to the GCKS element, that is responsible for their implementation. On the framework represented in Figure 11, the Policy Server interacts only with the GCKS. However [24] also foresees interactions with other elements of the reference framework.

The Sender and Receiver elements belong to the data handling functional area. They are concerned with security-related data transformation such as data encryption, and data authentication (source or group authentication[8]). The GCKS element is responsible for the Group Key Management functional area. This area includes the management of group cryptographic keys, their state, and other relevant key related information, normally described as SAs. The Multicast security policies functional area, implemented by the Policy Server element, defines the rules of operation of the all other elements in the framework. Despite the possibility of individual distribution of policies to all elements, group policy distribution provides an high level of coordination. Group policies or group context management work can be found in [32] and [33].

---

[8] Group authentication should be understood as a guarantee that the multicasted data was originated, or last modified, by a member within the secure multicast group.

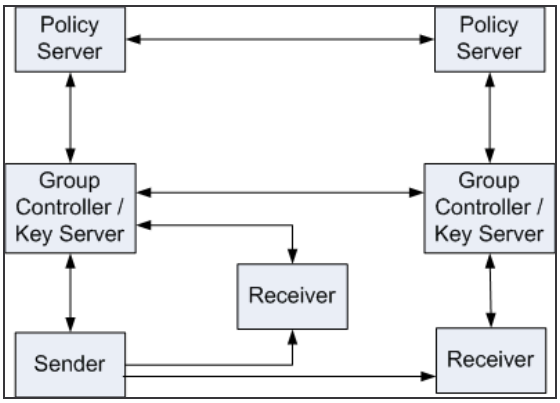

Figure 12 - Distributed Multicast Security Reference Framework

Scalability arises from the need of supporting large secure multicast groups over wide areas or even global secure multicast groups. The MSEC approach to the scalability problem of its centralized framework resides in the distribution of its elements, as can be observed in Figure 12. Thus, GCKS elements can interact with other GCKS elements in order to support large amount of member requests, for example. Global membership can also be distributed by a set of GCKS elements. The distribution of the GCKS elements implies the distribution of Policy Server elements, in order to avoid bottlenecks.

In Figure 12, the Sender interacts with two Receivers, each one interacting with its nearby GCKS. Each element must peer with other elements of the same type, for information exchange and synchronization. Element authentication and authorization is required.

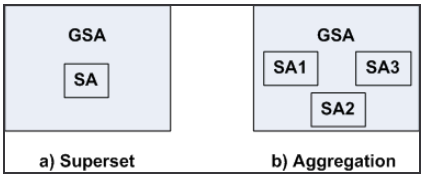

Figure 13 – Relationship of GSA to SA

A key concept defined by in the MSEC Secure Multicast Architecture is the Group Security Association (GSA), that is based on the concept of SA. A SA describes a set of policy and cryptographic keys which provide security services for the network traffic matching that policy. A SA contains the following attributes [24]:

- Selectors, such as source and destination transport addresses;
- Properties, such as a Security Parameter Index (SPI) or cookie pair;
- Cryptographic policy, such as the algorithms, key lifetimes, and key lengths used for authentication or confidentiality;
- Keys, such as authentication, encryption and signing keys.

Besides the information contained in the standard SA, GSA also contains information required for securing multicast scenarios. A GSA can then be built as an aggregation of SAs (Figure 13.b) or as a superset of a SA (Figure 13.a). As a superset of a SA, the GSA must contain the group attributes and the group policies, such as the events triggering rekey[9] operations (member join, member leave) or group credentials. A superset of SAs is a set of independent SAs. The categories of SAs in this case are:

- Registration Security Associations – Unicast SA from a member to the GCKS. One per member.
- Rekey Security Associations – Multicast SA from GCKS to all group members. One per group.
- Data Security Associations – Multicast SA from Senders to all group members. One per Sender.

4.2.2. Key management

Multicast is a scalable form of group communications. Secure multicast communications are also expected to be scalable, despite the difficulty of distributing cryptographic keys. These can be overcome by using three approaches [34]: 1) centralized group key management protocols; 2) distributed key management protocols; 3) decentralized architectures. The first is characterized by the existence of a unique entity with the responsibility of managing the entire group; it is focused in bandwidth usage optimization, key storage minimization, and reduction of computational power. The second assumes that any member can be a key distribution server and perform access control; all members can contribute to the generation of the group key. The decentralized architecture approach achieves scalability through the division of the global group in a set of subgroups, each one having its own subgroup manager.

Some proposals arose for the decentralized architectures approach; they include the Scalable Multicast Key Distribution (SMKD) [35], the Iolus framework (Iolus) [36], the Dual-Encryption Protocol (DEP) [37], the Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences (MARKS) [38], the Cipher Sequences (CS) [39], the Kronos approach (Kronos) [40], the Intra-Domain Group Key Management Protocol (IGKMP) [41], and the Hydra decentralized group key management architecture [42].

---

[9] Rekey means the operation of distributing a new cryptographic key that substitutes the current key used to decrypt the group communications.

The evaluation of the efficiency of these schemes in decentralized frameworks can be made based on factors such as key independence, local rekey, and rekey per membership. Other attributes that can be used include the type of communication and the use of decentralized group controllers. Key independence is related to the past distributed keys not being compromised upon a key disclosure and thus protecting previous group communications from access by a latter joined element that as saved past group communications. Local rekey is related to the number of elements affected by a group size change; that is, if there is a departure from the group, it should be a local departure and affect only the elements within the same local subgroup, avoiding scalability problems such as the "1-affects-n" problem. Rekey per membership is related to backward and forward secrecy; in other words, if when a member joins a group it gets access to a key that will allow it to decrypt messages sent previously, then there is no backward secrecy; if after a member group leave operation it still can decrypt group messages, then there is no forward secrecy. The group communications are mainly of two types: groups having only one data source, and groups having multiple data sources. The use of decentralized controllers improves the availability of the service; the failure of a central group controller affects all group communications, and induces failures similar to those caused by the group key management protocols described earlier.

Table 3 compares the decentralized architectures described above. Kronos and MARKS do not have key independence, and future cryptographic keys can be derived from current key; in Kronos the new key is generated from old keys, and in MARKS new keys are generated from seeds that, if compromised, imply the failure of future secrecy. Iolus is the only one, supporting local rekey procedures. Its rekey operation involves only the members of a subgroup, and does not affect the other subgroups. Local rekey enables a certain degree of scalability and bypasses the "1-affects-n" problem, but it has visible implications in the data path. Translations are required in communications from one subgroup to another. Past and future secrecy can be achieved by doing a rekey when a group membership changes. DEP, for instance, adopted a timed rekey that rekeys the group independently of membership changes but periodically, it enables periods of time during which the leaving members still can access the group data, that is, there is no future secrecy in that time period. SMKD does not guarantee future secrecy at all, because it does not rekey with membership changes, and the new keys are distributed based on old keys. In DEP and CS there exists a central group controller entity that must be contacted on every group join authentication, despite of the existence of subgroups and subgroup managers, what creates a single point of failure.

	Key Independence	Local Rekey	Rekey per Membership	Type of Communication	Decentralized Management
SMKD	Y	N	N	m-to-n, 1-to-n	Y
Iolus	Y	Y	Y	1-to-n	Y
DEP	Y	N	N	m-to-n, 1-to-n	N
CS	Y	N	Y	1-to-n	N
MARKS	N	N	N	m-to-n, 1-to-n	Y
Kronos	N	N	N	m-to-n, 1-to-n	Y
IGKMP	Y	N	Y	m-to-n, 1-to-n	Y
Hydra	Y	N	Y	m-to-n, 1-to-n	Y

Table 3 – Decentralized architectures comparison table

The solution we propose in the next chapter fits into the class defined by the last of the three approaches presented. We will adopt a distributed architecture, and define subgroup managers (Multicast Deflectors) which are responsible for authentication, group management, key generation and key distribution on its network area.

5. Secure video distribution system

The aim of this chapter is to propose a solution for the scenario presented in Chapter 1. We start by introducing the system requirements. Then, we present the system specification. The design section follows and, there, we describe the architecture of all the elements and their functionality. The application protocols are presented next. Concluding this chapter, we discuss the innovation of the solution proposed.

5.1. System requirements

In order to describe the system requirements and to help the development of a solution, three scenarios were considered: 1) an unauthorized user tries to access a video channel; 2) an authorized user tries to access a video channel for which he has no permission; 3) an authorized user tries to access a video channel for which he has permission. All the scenarios assume the transmission of multiple video channels, and each channel can be viewed only by authorized users; when the channel is not visualized by any user in some last mile network area, it shall not be transmitted. Based on that we identified the following requirements:

1. Encrypted data transmission – video channel streams must be encrypted prior to their transmission;

2. Unique user identification and authentication – each user must be identified individually in order to authenticate itself and enabling access control;

3. Unique channel identification – it must be possible to distinguish each channel;

4. Access control by user and channel – it must be possible to prevent an authenticated user from accessing channels for which it has no authorization;

5. Channel source authentication – it must be possible for a client to be sure that it is receiving data from the correct server;

6. Group transmission – each video channel should be transmitted only once for all the interested clients;

7. Transmit only if required – if there are no clients interested in a video channel it shall not be transmitted;

8. Restringe user mobility – a set of user credentials should only be valid in the user's network area;

9. Scalability – is should be possible to support as many users as required.

The usage of encrypted data transmissions (requirement 1) prevents unauthorized video channel access; it depends on the authentication of receivers and on the encryption of the transmitted video channels. Requirement 6 may enable network bandwidth reduction since the video channels are sent to groups of users (one copy per group of users), and not to individual users (one copy per user). Both requirements can be fulfilled through the usage of secure multicast transmission of simultaneous video channels. The requirement 4 can also be helped by secure multicast transmission, since it may enable individual user identification, authentication and access control. This is particularly true when each transmitted video channel is translated into a new secure multicast session that uses a multicast group access control algorithm, as those described in [36] and [43]. In these forms of group authentication, each user receives periodically a cryptographic key which enables him to access the video channel encoding cryptographic key; this channel key is encrypted with another key that was initially negotiated with the GCKS (session key) and transmitted to the user.

The individual identification of either users or channels (requirements 2 and 3) can be met by providing to each user and to each channel a unique reference or number. In order to support user authentication (requirement 2), conventional encryption can be adopted and a cryptographic key can be pre-shared and associated with the users individual reference. Hence, if a message is received and it carries out a user reference in clear text, then the authentication process can select the user's pre-shared key and decode the encrypted portion of the message that also contains the user's reference; if both references match, the user is authenticated. Authenticating the video channel transmission source (requirement 5) will protect users from receiving video channels which are not original; either partially tampered

or totally modified video channels. This requirement may suggest the adoption of asymmetric cryptography techniques. Although adequate to this case, asymmetric cryptography introduces significant delays in the video channel transmission when compared to conventional, symmetric, encryption. This may lead to the selection of the latter.

On group communications the data is sent to the group and not to the user, thus reducing the number of data connections on the network. This is true on a multicast enabled core network environment but not in the last mile network area composed by OSI layer 2 broadcast media. On the other hand, if there is no user interested in a video channel, this channel shall not be transmitted in that network segment (requirement 7). By avoiding the transmission of unnecessary video channels, it enables the medium to be available for other uses such as standard Internet access.

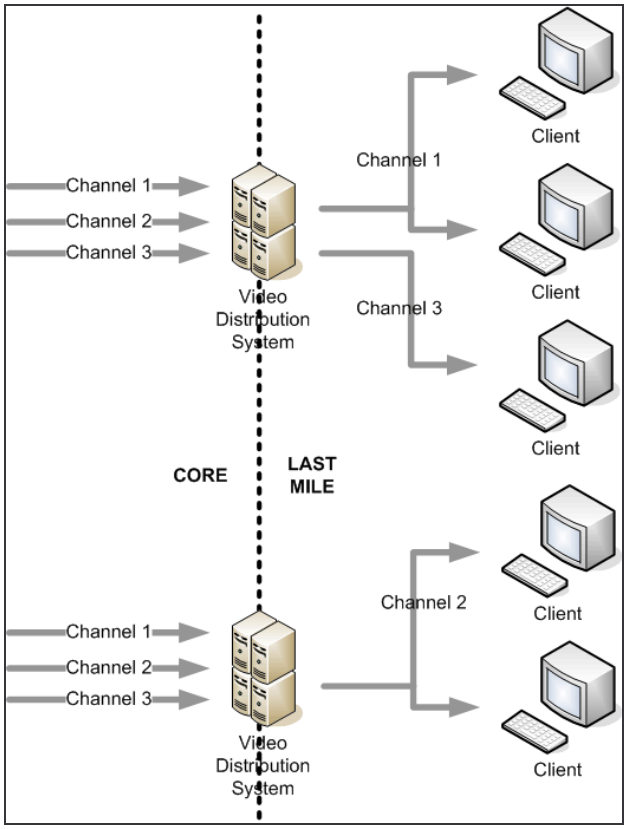

Figure 14 – Video distribution system

The solution proposed addresses scalability (requirement 9) using the same solution of the MSEC reference framework (see Figure 14); MSEC approaches scalability through Policy Server and GCKS elements distribution, forcing group management information and group policy synchronizations and introducing replication mechanisms. As a consequence, we gain the support of user mobility. User mobility is not demanded and will not be provided by the proposed solution (requirement 8); the synchronization / replication mechanism foreseen by the MSEC group will not be used also. The local equipment of the video distribution system

will be responsible for the client authentication and it will be the element having the client credentials. The result of this authentication model is a reduction of local group size, which improves the scalability of the proposed solution [44].

5.2. Video distribution elements

Figure 15 depicts the elements involved in the proposed video distribution system: the Multicast Deflector (MD), the Sender and the Receiver.

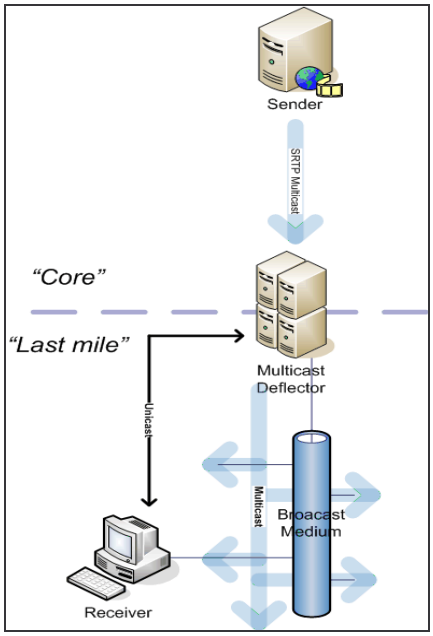

Figure 15 – Secure video distribution system components

The secure video distribution system proposed is heavily based on the MSEC reference framework. Our Receiver element is analogous to the Receiver element of the MSEC framework; our Sender element is similar to the MSEC Sender. The group key management functional area of the MSEC framework is also implemented in our MD element; MD has the same responsibilities of the MSEC GCKS that include user authentication, user authorization, key distribution and generation. But it also includes some functions hold by the MSEC Policy Server, such as GSAs or rekey interval definition.

The Sender element is responsible for generating the IP packet stream of one or more video channels. If this element resides in the service provider core network, then Real-time Transfer Protocol (RTP) can be used to transport the video channel to the MD element; on the other hand, if the Sender is placed anywhere on the Internet, then Secure Real-time Transfer Protocol (SRTP) [45] must be used to transmit the video stream in order to prevent unauthorized access. In a scenario of multiple video channel distribution, several Sender

elements are expected to arise and, despite the original video channel streams being RTP or SRTP, the streams sent in the last mile network must always be transported in SRTP streams.

The main function of the Receiver element is to display video channels. In order to do this it must decode the SRTP streams sent in the last mile network. The Receiver element shares a cryptographic key with the MD what enables it to acquire Session Encryption Keys (SEKs) and to authenticate itself before the MD element. The SEKs are then used by the Receiver to decrypt messages containing the Channel Encryption Key (CEK), which will enable the Receiver to access each video channel SRTP streams.

The MD is the key element of the proposed architecture; it is responsible for (1) group management, (2) cryptographic key generation and distribution, (3) Receiver authentication, (4) (re)encryption of video channel streams with SRTP, (5) transmission of requested video streams, and (6) reception of all the video channel streams coming from the senders. Group management is achieved through our Implicitly Managed Group Protocol (IMGP) (not IGMP) and it consists on the creation of a layer 4 multicast group per video channel. When a Receiver wants to view a channel, it requests it to the MD and, on a successful reply, it becomes part of that channel group for a limited period of time. The MD authenticates Receiver elements by means of a pre-shared cryptographic key. In other words, if the MD can decrypt correctly the Receiver requests, it means that the correct pre-shared key was used and that the Receiver identity was verified. All video channels must be received by the MD; this decision enables fast responses to Receivers requests as well as filtering of video channel transmissions in the last mile network. The MD will only transmit the channels corresponding to the groups having one or more Receivers subscribed. Last mile network area access control is achieved be encrypting the video streams with SRTP.

Multiple levels of network optimizations are accomplished with our video distribution system. At the service provider core network the optimization is achieved by adopting multicast transmission. In the last mile network, considering broadcast media, multicast may not perform the same level of optimization. In this case, optimization is obtained by limiting the channels transmitted in the last mile, to the video channels requested, in conjunction with standard multicast transmission. All the streams are sent to a unique multicast address, simplifying other network functions such as the definition of QoS. The video channels differentiation is made by port numbers, what makes our solution a layer 4 video distribution solution.

Multicast traffic filtering at the MD element is based on IP routing configuration. The two multicast addresses used, one for the core network, and the other for the last mile network, are associated to different network interfaces. Filtering is made by not forwarding IP traffic. With

this configuration, the core network multicast streams are forced to end in the MD, the multicast streams generated by the MD are sent to the last mile network interface.

Communications between the MD element and each Receiver element involve several network flows: one unicast and, at least, two multicast flows. The unicast flow is used for individual communication; each Receiver uses it for operations such as group joining (i.e. video channel request); exchange SEKs and group rekey operations. Two multicast flows are used per channel by the MD to communicate with the Receivers. One is used to exchange CEKs, and the other to exchange the video channel streams.

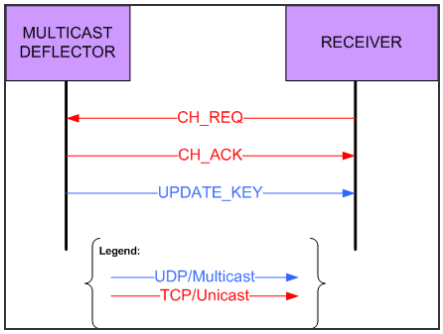

Figure 16 – Simple channel request

User access control is made through the use of cryptographic techniques, where each multicast video stream transmitted to the last mile network is encrypted with a unique cryptographic key that is defined as a CEK. The Receiver element needs to send a channel request to the MD in order to obtain a new SEK that is used to decrypt the multicast messages containing the CEKs. Considering the possibility of existing more than one Receiver interested in the same video channel, the CEK is transmitted to all Receivers simultaneously. This is done by appending several copies of the CEK and by encrypting each copy with a different SEK. When a Receiver gets a CEK message it searches for the part that is encrypted with its SEK and decrypts it.

5.3. System design

The complete secure video distribution system architecture is shown in Figure 17, where all the modules of all elements are depicted. The key element MD is explained next, followed by the Receiver element.

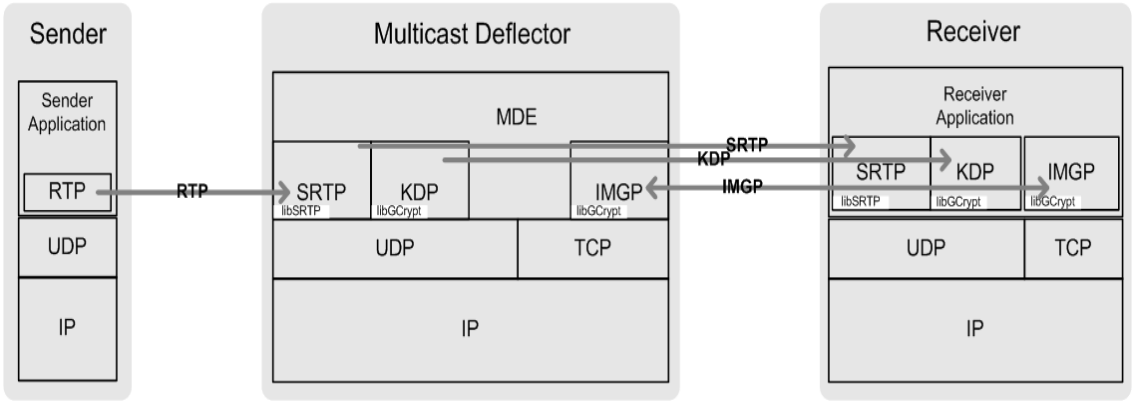

Figure 17 – Secure video distribution system architecture

5.3.1. Multicast Deflector

The MD Architecture is depicted in Figure 18, and it consists of 4 main modules: Multicast Deflection Engine (MDE), SRTP, KDP, and IMGP.

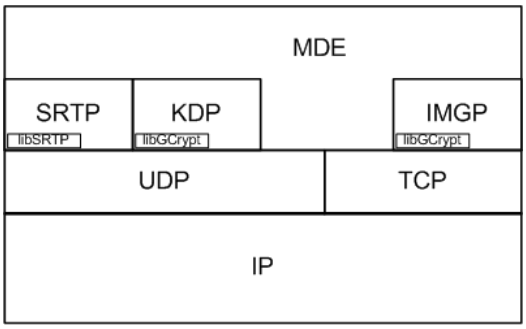

Figure 18 – MD architecture

The MDE is the controller module. It generates new cryptographic keys (SEK or CEK) upon IMGP requests, instructs KDP to distribute CEK to Receivers, configures the IP routing of the MD, and launches the execution of the SRTP, KDP and IMGP modules. The MDE is also responsible for the authorization of Receiver channel requests, for instructing the SRTP module to start or stop transmitting video channel SRTP streams and distributes CEKs. Group management is another responsibility of the MDE, and it is done by maintaining lists of channels, each one with its list of currently subscribed Receivers. Group management is the main function of the MDE.

The SRTP module is a simple RTP / SRTP receiver and SRTP sender. This module receives the streams sent by the Sender, either RTP or SRTP, and encrypts them with the corresponding CEK into new SRTP streams sent to the last mile networks.

The IMGP module implements the IMGP protocol. This protocol is responsible for transporting the SEK of each receiver, when it requests the reception of a video channel. Receivers must send an encrypted channel request message (CH_REQ message) to the MD,

which will answer with and acknowledge or not acknowledge message, depending upon the acception of the request. The acknowledge message includes the new SEK, that will be used to encrypt the messages exchanged in the Key Distribution Protocol (KDP). The IMGP messages are encrypted with the Receivers pre-shared encryption key, enabling data confidentiality and Receiver authentication.

The KDP module implements the KDP protocol that allows CEK distribution to all SEK owners by sending the messages containing the CEK (UPDATE-KEY message) in multicast to all local Receivers. Each secure video channel stream has one, and only one, CEK at time, and it must be delivered to all authorized Receivers interested in that video channel. CEK keys are generated at the MD upon the first IMGP request for that video channel, and subsequent requests will receive the same CEK. When the CEK Time-To-Live (TTL) is reached, a new rekey process will occur with the first channel request reception, and a new CEK will be generated and sent in multicast. The UPDATE-KEY message contains several copies of the CEK, one for each interested Receiver and encrypted with its SEK.

Communication between modules is IP based, some cases using UDP, and TCP in the others. TCP connections are adopted for video channel requests due to their importance. The channel request must be reliable. The communications inside the MD element, i.e. between the MD modules, can be seen as a low error communication, and there is no need for TCP connection management overload; UDP is used, in alternative. For multicast communications UDP is a must. No cryptographic mechanisms were adopted in the communication inside the MD element, since it is considered reasonably safe from eavesdropping.

5.3.2. Receiver

The Receiver architecture is depicted in Figure 19 and it has 3 modules: SRTP, KDP and the IMGP module. These modules will act as clients of the homonym modules at the MD element. The SRTP module is mainly responsible for decrypting video channel SRTP streams, using the corresponding CEK. The CEK is obtained by the KDP module, provided the channel request has been processed correctly by the MD. The IMGP module enables the Receiver to send video channel requests.

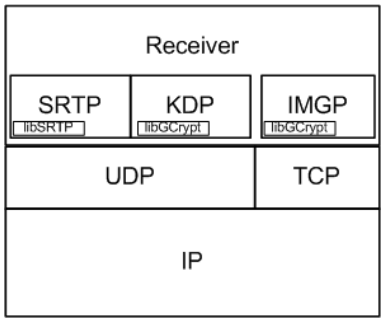

Figure 19 – Receiver architecture

5.3.3. Message sequence charts

The messages exchanged between all elements, including the modules of the MD and the Receiver elements, for a scenario of successful video channel requests, is shown in the Message Sequence Chart (MSC) of Figure 20. As described earlier, there is the need for at least two multicast flows and one unicast flow, in order for the receiver to successfully request a video channel on its last mile network.

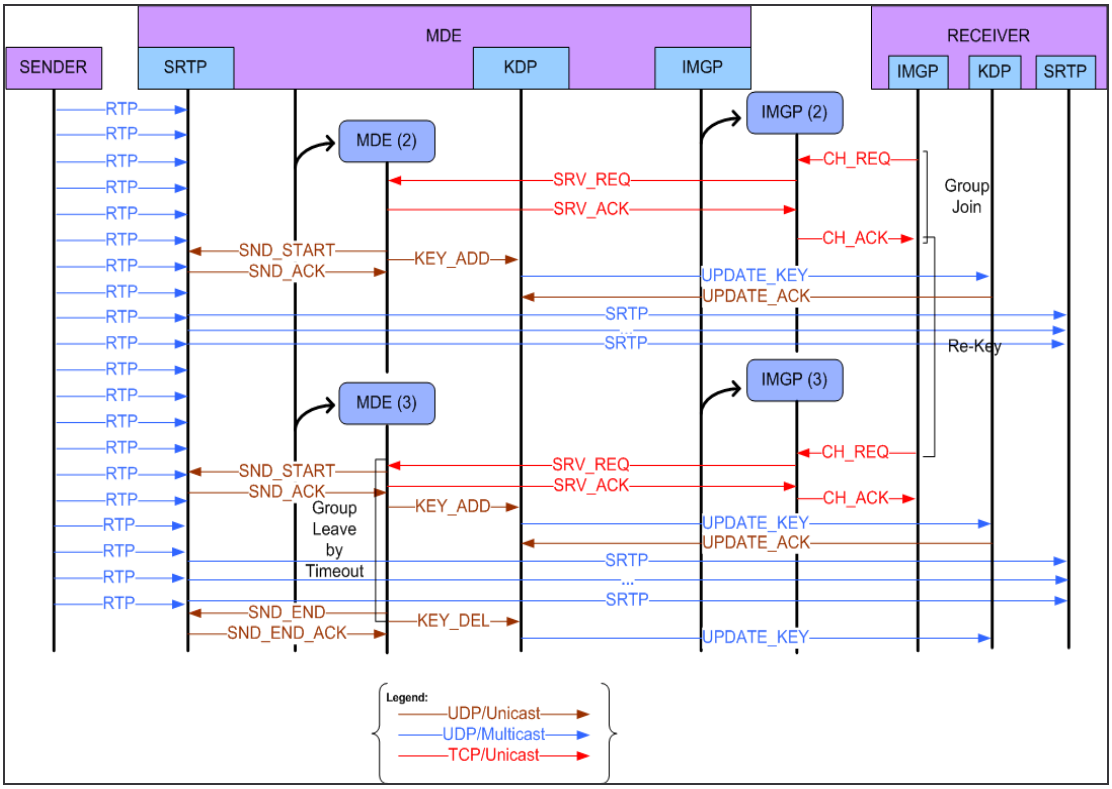

Figure 20 – Successful channel request

Communication is always started by the Receiver, which uses the IMGP protocol to request a video channel through a TCP connection to the port 3770 of the IMGP module at the MD. When the IMGP module at the MD receives a new TCP connection, it generates a new instance of itself to process Receiver requests, until that connection is closed. The Receiver sends the CH_REQ (channel request) message to the MD, which is then processed by the

newly created instance of the IMGP module. The IMGP module, in turn, sends the SRV_REQ message (service request) to the MDE module. Then, the MDE module validates the receiver identity and request and, if all is correct, the MDE generates the CEK and the Receiver SEK. It also instructs the SRTP module to start transmitting the channel SRTP stream encrypted with the CEK and instructs the KDP module to start the key update process. At the end of the Receivers request process, the TCP connection to the MD's IMGP module is closed, thus terminating that IMGP instance. The key update process allows the CEK multicast transmission to all the Receivers that requested that video channel. By receiving the CEK, the Receivers become able to decrypt the video channel SRTP stream.

The CEK generated by the MDE has a limited TTL. The main purpose of this TTL is to ease the management of interested Receivers; if no new CH_REQ is received during the last TTL then the Receiver is removed from that channel group. If the Receiver intends to continue receiving a video channel, then it must renew the channel request by sending a new CH_REQ message to the MD element.

The packet loss probability in communications between modules of the MD is very low because these are internal communications. But when considering communications between de MD and the Receivers, this probability may increase. In Figure 21, the communication example describes a failure in the transmission of the UPDATE-KEY message, which is sent over UDP to a multicast group. A failure in this message disables the Receiver from decrypting the video channel stream, because it will not have access to the new CEK. The Receiver, after receiving the channel request acknowledge message (CH_ACK), starts a timeout and if the UPDATE_KEY message is not correctly received from the MD, it sends a UPDATE_NAK message to the KDP module at the MD, so the latter can resend a Receiver specific UPDATE_KEY message, containing only its CEK. The timeout control between modules is based in shared variables, such as the CEK variable, shown in Figure 21.

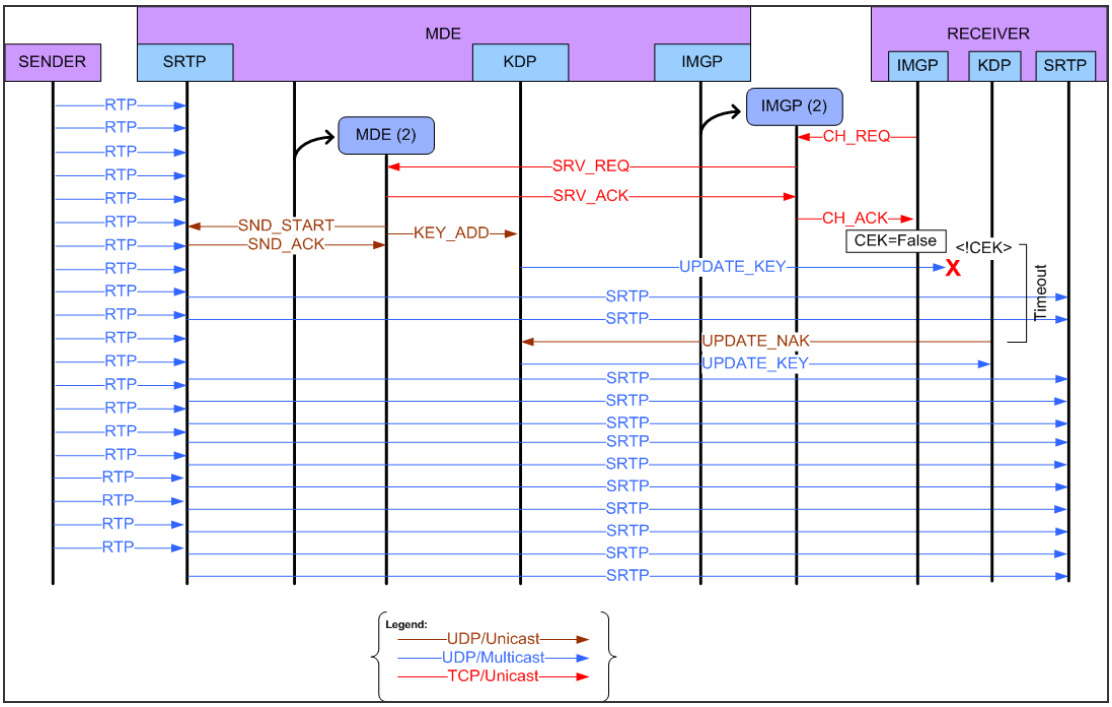

Figure 21 – Failed channel request caused by a failure in the UPDATE_KEY message

Another possible point of failure is the KEY_ADD message that is sent from the MDE module to the KDP module, so that the KDP can include the Receivers CEK encrypted with the Receivers SEK in the UPDATE_KEY message. If the transmission of the KEY_ADD message fails (Figure 22), or if this message is only sent by the MDE module after the KDP has sent the UPDATE_KEY message, the client may receive UPDATE_KEY messages and SRTP packets from the video channel stream but, without the correct CEK, it will not be able to decrypt the SRTP packets. However, it still can process UPDATE_KEY messages, because it has a valid SEK; but no UPDATE_KEY message will contain that Receiver's CEK for that specific channel. In this case, the Receiver resends the channel request message. Again, the synchronization of the 3 modules at the Receiver is based on shared variables. Setting the REPEAT variable to true forces the repetition of the channel request.

A channel request not acknowledged is represented in the Figure 23. In this case, the Receiver issues a CH_REQ message to the IMGP module that, in turn, issues a service request (SRV_REQ) to the MD main module, which verifies that the Receiver is not authorized to access the channel requested. It replies with an not acknowledge message (SRV_NAK) to its IMGP module that, in turn, sends an not acknowledge channel request message (CH_NAK) to the Receiver, notifying it about the service failure, and allowing it to execute a new channel request. If all goes well, the Receiver should receive its CH_ACK message, confirming the new channel availability and obtaining the Receivers SEK.

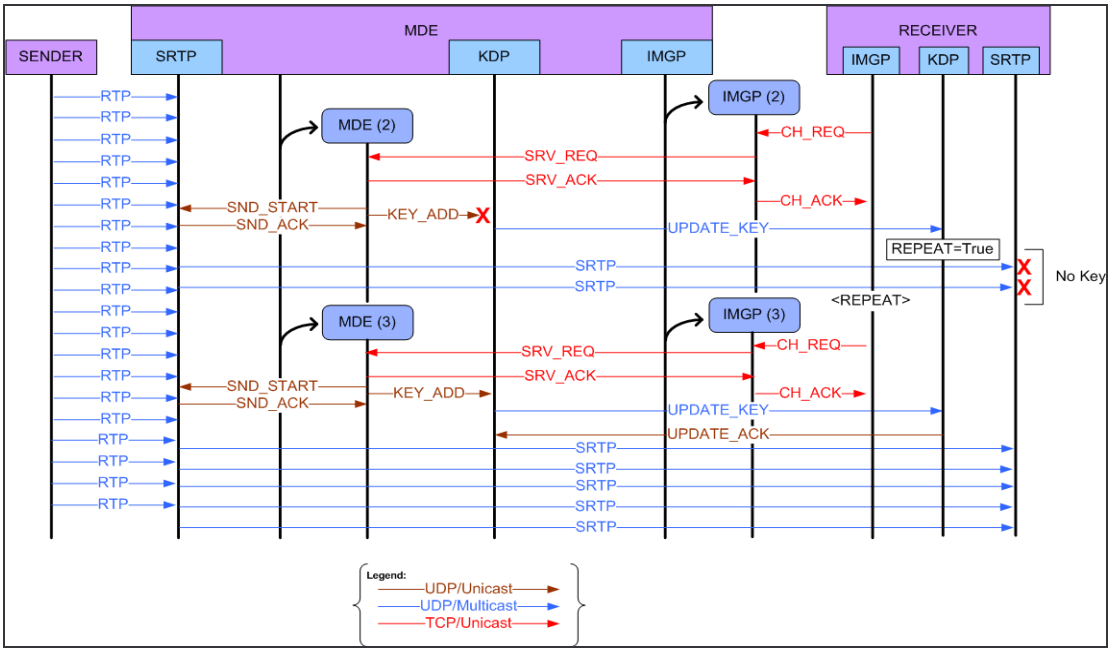

Figure 22 – Failed channel request caused by a failure in the KEY_ADD message

In the MSCs presented, we can observe three types of flows used by the MD and the Receiver. These flows are (1) one unicast TCP flow between the IMGP modules in the MD and the Receiver, for SEK exchange and group management; (2) one multicast flow between the KDP modules for CEK exchange; and (3) a multicast flow used by the SRTP modules for video stream distribution.

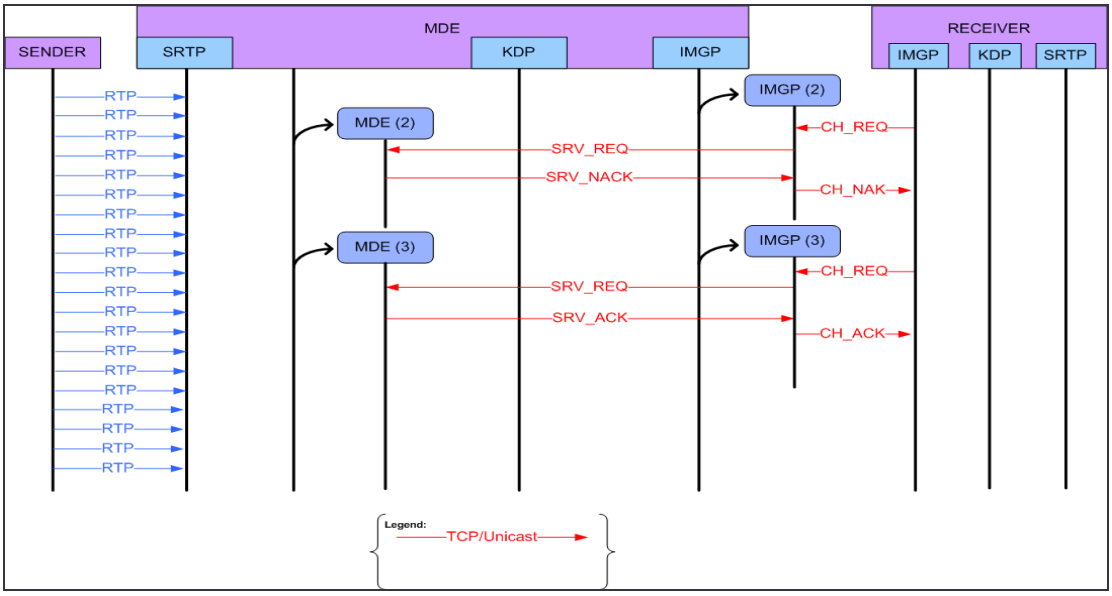

Figure 23 – Not acknowledge channel request

5.3.4. Implicitly Managed Group Protocol

The IMGP is a simple protocol which manages groups of users interested in a channel and accepts user channel requests. The group join operation is made by the CH_REQ message

(Figures 23 and 24), and is confirmed positively through an acknowledge message (CH_ACK), or negatively through a not acknowledge message (CH_NAK). Group leave operations are implicit, that is, they are based in timeouts and, therefore, use no protocol messages.

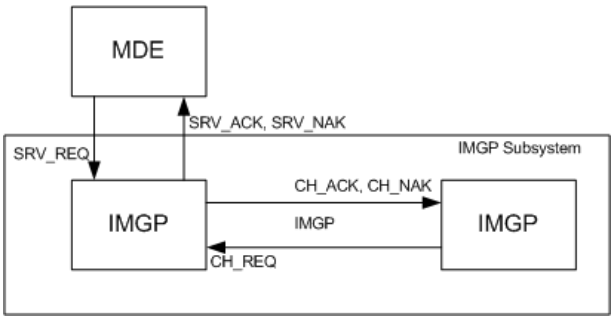

Figure 24 – IMGP subsystem

The IMGP module at the MD interacts with its peers at the Receivers. It accepts channel requests, forwards them to the MDE using the SRV_REQ message, and waits for the confirmation, which can be received through an acknowledge message (SRV_ACK), or a not acknowledge message (SRV_NAK).

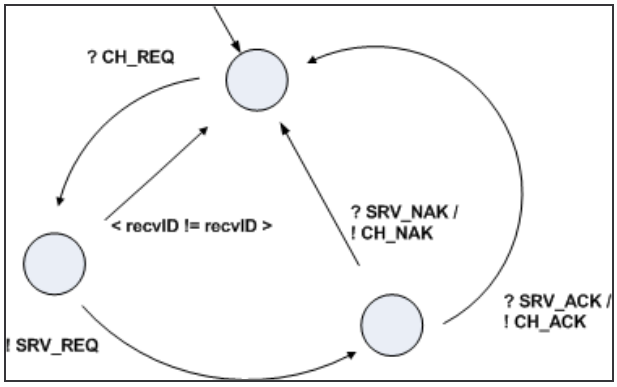

Figure 25 – IMGP state machine at the MD

As depicted in Figure 25, the IMGP module at the MD first waits for a video channel request from a Receiver; upon its reception it sends a service request to the MD and waits for the MD response. On a positive response the IMGP sends a channel request message confirmation; on a negative response it sends a not acknowledged channel request. The video channel request message sent by the Receiver to the MD IMGP module is encrypted with a symmetric pre-shared encryption key that enables the confirmation of the Receiver identification.

The channel request message (CH_REQ) sent by Receivers is represented in Figure 26 and it consists of three fields: 1) a message type identification string; 2) the Receiver identification code; and 3) and encrypted field. The encrypted field contains a message identification sequential number, the requested channel identification code, and the Receiver identification code again. This part is encrypted with the Receiver's symmetric cryptographic key that is

pre-shared with the MD element. The clear text part of the channel request enables the MD to understand that it is a channel request and to decide what cryptographic key to use. The encrypted part of the channel request enables the control of the message sequence and the confirmation of the Receiver identification, by comparing the clear text Receiver identification code (recvID) with the decoded Receiver identification code.

$$\text{CH\_REQ} : \left[\text{'REQ'}; \text{recvID}; \{\text{msgID}; \text{chanID}; \text{recvID}\} \text{K}_{\text{RECV}} \right]$$

Figure 26 – Channel request message

The channel request message confirmation (CH_ACK) sent by the MD to Receiver (Figure 27) is composed of four parts: 1) a message type identification string; 2) the message sequence number (plus one that the sequence number in the CH_REQ message); 3) a digital signature; and 4) an encrypted part. The encrypted part is composed of the new SEK that the Receiver must use to decrypt future messages sent from the MD element, and a time to live for that SEK. The signed part of the message enable Receivers to validate the MD identity; Receivers must be able to generate the same signature using the MD public key.

$$\text{CH\_ACK} : \left[\text{'ACK'}; \text{msgID}; \{\text{msgID}; \text{mdID}\} \text{K}_{\text{MD}} \textit{Signed}; \{\text{SEK}_{\text{RECV}}; \text{timeTL}\} \text{K}_{\text{RECV}} \right]$$

Figure 27 – Channel request acknowledge message

The channel request not acknowledge message (CH_NAK) is composed of three parts: 1) a message type identification string; 2) the message sequence number (plus one that the sequence number in the CH_REQ message); and 3) a digital signature. The digital signature allows the verification of the MD identity by Receiver elements. The CH_NAK message structure is depicted in Figure 28.

$$\text{CH\_NAK} : \left[\text{'NAK'}; \text{msgID}; \{\text{msgID}; \text{mdID}\} \text{K}_{\text{MD}} \textit{Signed} \right]$$

Figure 28 – Channel request not acknowledge message

The messages exchanged by the MDE and IMGP module are shown in Figure 29. The Receiver identification validation is responsibility of the IMGP module. The channel access authorization, made after identification verification, is transferred to the MDE in the form of a service request message (SRV_REQ). As a reaction to this service request, the response will be an service acknowledge message (SRV_ACK), except if the MD is unable to provide the video channel requested; on this case it replies with an not acknowledge message (SRV_NAK). No encryption mechanism was adopted in communications between modules at the MD.

$$\begin{aligned}&\text{SRV\_REQ}:\left[\text{'REQ'; msgID; recvID; chanID; SEK}_{\text{RECV}}\right]\\&\text{SRV\_ACK}:\left[\text{'ACK'; msgID; timeTL}\right]\\&\text{SRV\_NAK}:\left[\text{'NAK'; msgID}\right]\end{aligned}$$

Figure 29 – MDE-IMGP exchanged messages

Any receiver group leave operation is based in TTL (timeTL) expiration, allowing simple group management. The group leave operations force the update of the SEK lists in the last mile network areas transmitting the channel. The SEK list per channel, active in last mile area network is updated upon channel requests (SRV_REQ reception), by adding keys to the corresponding list, and on timeout, by removing keys. These SEK lists are used by the KDP.

5.3.5. Key Distribution Protocol

The KDP is also a simple protocol that allows the distribution of CEK to authorized receivers. It is based in the key distribution messages adopted in Iolus framework [36] and in the protocol described in [43] . The KDP subsystem is shown in Figure 30, where four message types can be identified: 1) KEY_ADD; 2) UPDATE_KEY; 3) UPDATE_ACK; and 4) UPDATE_NAK. Despite KEY_ADD messages are not part of the protocol, these have a significant role in the operation of the KDP module at the MD. This message instructs the KDP module to append a new SEK to the list of SEK of a channel and to initiate the key update procedures towards the Receivers.

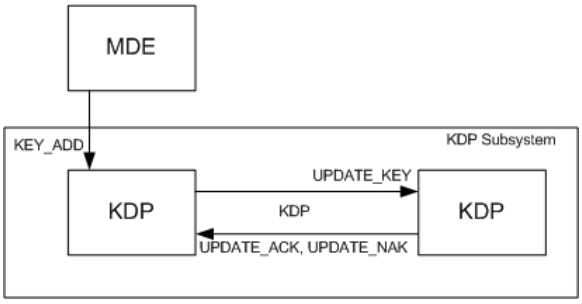

Figure 30 – KDP subsystem

The KEY_ADD message is sent by the MDE module to inform the KDP module about the cryptographic key of a channel (CEK) or about the cryptographic key of a Receiver (SEK). This message structure is shown in Figure 31 and it consists of four fields: 1) a message type identification string; 2) a numeric code representing the type of key (0 for CEK, other for SEK); 3) an identification code used for channel identification or Receiver identification, depending on the value of the previous field; and 4) the cryptographic key.

$$\text{KEY\_ADD}:\left[\text{'ADD'; keyTYPE; keyID; keySTR}\right]$$

Figure 31 – KEY_ADD message

The operation of the KDP module per channel is shown in Figure 32. Upon the correct reception of a KEY_ADD message, the cryptographic key in the message is validated by checking the value of the keyTYPE field. If the field has a value of zero, then it indicates that the message contains a CEK that must be stored for future use; if this field has a value other than zero, it indicates a Receiver SEK that will be used in the UPDATE_KEY (Figure 33) message construction. When a SEK is received, the KDP builds an UPDATE_KEY message based on the keys associated with the current channel and on the amount of numKEYS, and sends the message to all interested Receivers, using multicast UDP/IP packets. After sending the UPDATE_KEY message, three events can be processed by the KDP module: 1) the timeout expires; 2) there is an UPDATE_ACK reception; and 3) there is a UPDATE_NAK message reception. The commonly expected action is the number 2), which occurs when a Receiver confirms the correct reception of the UPDATE_KEY. When there is an error and the Receiver does not get the UPDATE_KEY message, it should send the UPDATE_NAK message to inform the KDP module about it, forcing KDP to send a new UPDATE_KEY message. At this point, the KDP module waits for numKEYS acknowledge messages until a timeout.

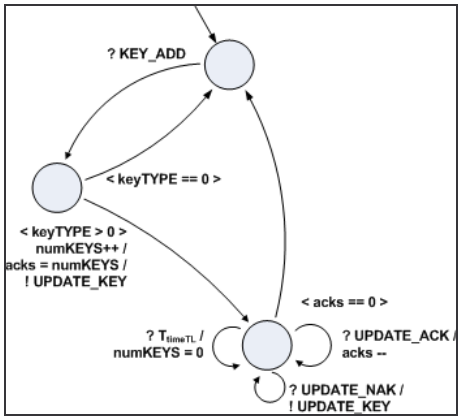

Figure 32 – KDP state machine

The UPDATE_KEY message has a variable size, and depends on the number of Receivers viewing the channel. This message is composed of five fields: 1) a message type identification string; 2) a message sequence number; 3) the channel identification code; 4) a digital signature; and 5) a variable set of pairs of Receiver identification codes and CEK encrypted with Receiver SEK. The number of pairs included in UPDATE_KEY messages depends on the scenario; if the UPDATE_KEY is generated by a KEY_ADD with a Receiver SEK or by an UPDATE_NAK message, then there will be only one pair in the UPDATE_KEY message; if it is generated as a consequence of a KEY_ADD message containing a CEK, then there will be a pair for each Receiver requesting the channel.

$$\text{UPDATE\_KEY}: \left[\begin{array}{l} \text{'UPD'; msgID; chanID; } \{\text{msgID; mdID}\} K_{MD} \textit{Signed}; \\ \text{recvID}_1; \{\text{CEK}\} SEK_{RECV_1}; \text{recvID}_2; \{\text{CEK}\} SEK_{RECV_2}; \\ \cdots; \text{recvID}_n; \{\text{CEK}\} SEK_{RECV_n} \end{array} \right]$$

Figure 33 – UPDATE_KEY message

The UPDATE_NAK and the UPDATE_ACK messages are identical in structure (Figure 34). They are composed of three fields: 1) a message type identification string; 2) the Receiver identification code; and 3) a field encrypted with the Receiver's SEK. The encrypted portion of the UPDATE_ACK and UPDATE_NAK messages consists of three fields: 1) the message sequence number; 2) the channel identification code; and 3) the Receiver identification code. If the clear text Receiver identification code equals the Receiver identification code decrypted with the Receiver SEK, then the identification of that Receiver is confirmed.

$$\text{UPDATE\_ACK}: [\text{'ACK'; recvID; } \{\text{msgID; chanID; recvID}\} SEK_{RECV}]$$
$$\text{UPDATE\_NAK}: [\text{'NAK'; recvID; } \{\text{msgID; chanID; recvID}\} SEK_{RECV}]$$

Figure 34 – UPDATE_ACK and UPDATE_NAK messages

The CEK are valid only for a certain period of time and the KDP module does not know this value. Therefore the KEY_DEL message is used to inform the KDP module to stop distributing a certain CEK. This message is also used by the MDE module to instruct the KDP module to stop including the SEK of a given Receiver in future UPDATE_KEY messages. As shown in Figure 35, the KEY_DEL message has three fields: 1) a message type identification string; 2) a key type identification code; and 3) the respective key identification code. A keyTYPE field with a value of zero means that the keyID refers to a channel identification code, and it leads to the removal of a CEK from the KDP module. A keyTYPE with a value other than zero means that the keyID refers to a Receiver identification code; in this case the keyTYPE will include the channel identification code.

$$\text{KEY\_DEL}: [\text{'DEL'; keyTYPE; keyID}]$$

Figure 35 – KEY_DEL message

5.4. Implementation

In this section, we present the essential aspects of the implementation. We will use the MSC represented in Figure 20 as our reference use case. The communication is started by the IMGP module at the Receiver that sends a channel request message - chReq (see Table 4) to the IMGP module at the MD. The authentication process lead by the IMGP module is based in conventional encryption and on pre-shared keys, as can be observed in the Table 5, which

presents some code of the *workClient* function; after decrypting the encrypted portion of the message, it is possible to obtain the Receiver ID contained in that encrypted portion of the channel request message. If it matches the clear text version of the Receiver ID (*sID*), that is also presented in the message, the Receiver is considered as authenticated. Note that the ID used to select de pre-shared key used to decrypt the request, is the ID shown in clear text.

```
1.     struct chReq {
2.            char msgTYP[4];
3.            int sID;
4.            int dataSZ;
5.            unsigned char * encData;   };
6.
7.     struct chAck {
8.            char msgTYP[4];
9.            int msgID;
10.           int ttl;
11.           int sigSZ;
12.           unsigned char * encSig;
13.           int dataSZ;
14.           unsigned char * encData;   };
15.
16.    struct chNak {
17.           char msgTYP[4];
18.           int msgID;
19.           int sigSZ;
20.           unsigned char * encSig;    };
```

Table 4 – IMGP Messages data types

After the correct reception of the channel request and the authentication of the Receiver, the IMGP module at the MD sends a service request to the MDE module. The MDE module is the controller module and has several functions, which include: CEKs generation, SEKs generation, start and stop secure channel streaming, and key distribution. When the MDE receives a service request from the IMGP module, it first checks if the Receiver is authorized to access the request channel; if so, it sends out messages to the other modules.

```
1.     void workClient (int sock) {
2.     ...
3.     recvIMGPReqPacket(sock,&request);
4.            s=getSubscriber(request.sID,mainSL);
5.            cleartext=decrypt(s->key,request.encData,request.dataSZ);
6.            curID=atoi(strtok(cleartext,":"));
7.            curChannel=atoi(strtok(NULL,":"));
8.            curMsgID=atoi(strtok(NULL,":"));
9.
10.           if(s->sID!=curID) errOut("Bogus ID.",310);
11.    ...
12.    }
```

Table 5 – Excerpt of *workClient* function, IMGP module at the MD

These messages, as represented in Figure 20, are twofold: 1) SND_START and 2) KEY_ADD. The SND_START message is used to instruct the Sender module to start a new channel SRTP stream, and contains the CEK to be used. The KEY_ADD message is used to transfer SEK and CEK to the KDP at the MD, so that it can start to send them to the

Receivers. This behavior is shown in Table 6, where the *sendKDPKeyCmd* and *sendKDPKeyCmd2*[10] are used to send three similar messages: the first is used to send the CEK to the KDP module (KEY_ADD); the second is used to send the CEK to the MD SRTP module (SND_START); the latter is used to send the Receiver's SEK to the KDP module at the MD.

```
1.     void* workClient(void* arg) { ...
2.     if(chReKey==1) {
3.            buildKDPCmdPacket(&chKc,"ADD",0,cID,chList[cID].chKey);
4.            sendKDPKeyCmd(chKc);
5.            sendKDPKeyCmd2(chKc, SND_ADDRESS, SND_PORT);
6.            }
7.     buildKDPCmdPacket(&subsKc,"ADD",cID,sID,request.key);
8.     sendKDPKeyCmd(subsKc);
9.     ...
10.    }
```

Table 6 – Excerpt of *workClient* function, MDE module at the MD

The data structure used for representing either SND_START or KEY_ADD message data is the same, because they are similar (Table 7); this *keyCmd* was also adopted for SEK transfers between the MDE and the KDP modules at the MD. These messages need the same information which consists of the channel ID (*keyCmd.id*) and the CEK (*keyCmd.key*). The field *keyCmd.type* becomes necessary for sending the Receiver's SEK; there is a need to identify both the channel and the Receiver, and the *keyCmd.type* is used for that purpose; *keyCmd.type* equal to zero means that the *keyCmd.id* contains a channel ID, while *keyCmd.type* larger than zero means that the *keyCmd.id* has a Receiver ID.

```
1.     struct keyCmd {
2.            char cmd[4];
3.            int type;
4.            int id;
5.            unsigned char key[BLOCKSIZE];
6.            };
```

Table 7 – SND_START and KEY_ADD messages data type

As decribed earlier, the module responsible for key distribution is de KDP at the MD, thus it must be aware of all CEKs and SEKs of all Receivers viewing the channels in order to build the KEY_UPDATE message, used to transfer CEKs to the active Receivers. Considering both the future possibility of allowing the Receiver to receive more than one channel simultaneously and the need for a structure adaptable to Receivers changes, the structure adopted, see Table 8, is both fixed and dynamic. Fixed because it uses a fixed size array to

[10] The *sendKDPKeyCmd2* message does not differ from the *sendKDPKeyCmd* in functionality. The *sendKDPKeyCmd2* enables the definition of the destination address of the message, instead of using default addresses.

represent the video channel list; typical providers supply 50 video channels that is far from the 1024 video channel limit imposed by this array.

```
1.     struct channelKeyList {
2.            int subsID;
3.            unsigned char key[BLOCKSIZE];
4.            struct channelKeyList* next;
5.            };
6.
7.     struct channelList {
8.            int on;
9.            unsigned char key[BLOCKSIZE];
10.           struct channelKeyList* chKList;
11.           };
12.
13.    struct channelList chList[1024];
```

Table 8 – KDP module data structures at the MD

Each element of the array contains the channel CEK, a field indicating if the channel is being transmitted (*channelList.on*) and a linked list (*channelList.channelKeyList*) containing the SEK of the Receivers which are viewing the channel (Figure 36). The use of this linked list enables a variable number of Receivers to view each video channel; it also eases the process of building the KEY_UPDATE message.

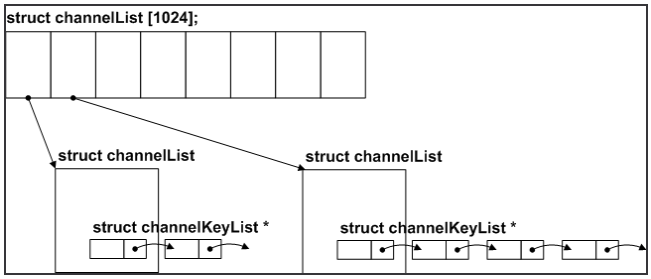

Figure 36 - KDP module data structures at the MD

The creation of the KEY_UPDATE packet is shown in Table 9, starting by the memory allocation process, followed by the filling of the main packet fields (message type, message ID, channel ID, MD's signature) and by the obtention of the pointer to the linked list of SEKs of the channel; it terminates in the encryption loop. This loop consists in adding, for each element of the linked list of that channel, a Receiver ID and a CEK encrypted with the Receiver's SEK.

```
1.     int sendKDPUpdPacket(struct updateKey* upd) {
2.     ...
3.     bytes=(sizeof(char)*4)+(sizeof(int)*5)+upd->sigSZ+(upd-
>nKeys*BLOCKSIZE);
4.     dgram=(unsigned char*)calloc(bytes,sizeof(char));
5.     memcpy(dgram,upd->msgTYP,4); offset=4;
6.     memcpy(dgram+offset,&upd->msgID,4); offset+=4;
7.     memcpy(dgram+offset,&upd->chID,4); offset+=4;
8.     memcpy(dgram+offset,&upd->sigSZ,4); offset+=4;
9.     memcpy(dgram+offset,upd->sig,upd->sigSZ); offset+=upd->sigSZ;
10.    memcpy(dgram+offset,&upd->nKeys,4); offset+=4;
11.    tmp = upd->chKList;
12.    for(i=0;i<upd->nKeys;i++) {
13.           memcpy(dgram+offset,&tmp->subsID,4); offset+=4;
14.           ciphertext=encrypt(tmp->key,chList[upd->chID].key,BLOCKSIZE);
```

```
15.          memcpy(dgram+offset,ciphertext,BLOCKSIZE); offset+=BLOCKSIZE;
16.          tmp=tmp->next;
17.          }
18.    sendto(sock,dgram,bytes,0,(struct sockaddr *)&target, sizeof(target));
19.    close(sock);
20.    free(dgram);
21.    return 0;
22.    }
```

Table 9 – Excerpt of *sendKDPUpdPacket* function, KDP module at the MD

The cryptographic functions used to encrypt and to decrypt use functions of a cryptographic library – the Libgcrypt. The encrypt function is shown in Table 10, where it obtains a context handle for the cryptographic module which, in this case, uses the AES encryption algorithm in the cipher feedback mode, defines the cryptographic key to be used, and initializes the vector for this encryption session. At the end, the data is encrypted and the result returned.

```
1.     unsigned char* encrypt(unsigned char* key,unsigned char* data, int
len) {
2.     ...
3.     err = gcry_cipher_open(&hd, GCRY_CIPHER_AES, GCRY_CIPHER_MODE_CFB, 0);
4.     if(err) errOut("Failed opening AES cipher...\n",0);
5.     err = gcry_cipher_setkey(hd,key,BLOCKSIZE);
6.     if(err) errOut("Failed setting the key...\n",0);
7.     err = gcry_cipher_setiv(hd,iv,BLOCKSIZE);
8.     if(err) errOut("Failed setting the initialization vector...\n", 0);
9.     ciphertext = (unsigned char*) malloc(len*sizeof(char));
10.    err = gcry_cipher_encrypt(hd,ciphertext,len,data,len);
11.    if(err) errOut("Encryption failed...\n",0);
12.    ...
13.    }
```

Table 10 – Excerpt of *encrypt* function

5.4.1. Configuration files

All the elements of our architecture are configured using text files. Each module at the MD has its own configuration information, but some of that may be shared with other modules. Examples of configuration modules are given in Table 11 and 12, which present a list of subscribers and a list of channels, respectively.

```
1.     subscriber:12345:54321:6DDB166CDF35ED1C585098E6A4D885EB:1 2 3 4 5 6 7
8 9 10 23 54 45 87
2.     subscriber:12346:55321:A48B2D2F878411F27AE4E645BE51C229:1 2 3 4 5 6 7
8 9 10 23 54 45 87
3.     subscriber:12347:56321:3C7B98C95DF0D54109AE8CF0FEF6FADA:1 2 3 4 5 6 7
8 9 10 23 54 45 87
4.     subscriber:12348:57321:8B1B15C5C9970B213BB52F6C2F20AD8B:1 2 3 4 5 6 7
8 9 10 23 54 45 87
5.     subscriber:12349:58321:31F29FC5DB3F48C79B7F62CA6D7BD1A1:1 2 3 4 5 6 7
8 9 10 23 54 45 87
```

Table 11 – Subscriber list

A line of the subscriber list (Table 11) represents a Receiver and holds the information required about that Receiver. The line starts with the string “subscriber:” followed by the Receiver ID, the message ID, the Receiver pre-shared key, and the authorized channel list for that Receiver. In Table 12, each line represents a channel and it consists of the channel ID, the

core network stream source multicast address, the core network stream source port, the local SRTP stream multicast address, and the local SRTP stream port number.

```
1.     1 229.0.0.1 9001 230.0.0.1 9001
2.     2 229.0.0.1 9002 230.0.0.1 9002
3.     3 229.0.0.1 9003 230.0.0.1 9003
4.     4 229.0.0.1 9004 230.0.0.1 9004
5.     5 229.0.0.1 9005 230.0.0.1 9005
```

Table 12 – Channel list

The Receiver has also a configuration file, shown in Table 13. The parsing of the Receiver configuration file is based on tokens. For instance, when the word 'ID' is detected at the beginning of a line, it means that the Receiver ID follows. Other tokens used are the msgID for configuring the current message ID, leaderID for the MD ID, key for its pre-shared key, chList for the list of authorized channels, and the pair leaderIMGPAddr and leaderIMGPPorto for configuring the address of the IMGP module at the MD.

```
1.     ID 12346
2.     msgID 55321
3.     leaderID 10000
4.     key A48B2D2F878411F27AE4E645BE51C229
5.     chList 1 2 3 4 5 6 7 8 9 10 23 54 45 87
6.     leaderIMGPAddr 192.168.1.1
7.     leaderIMGPPort 8770
```

Table 13 – Subscriber configuration

5.4.2. Development platform and programming libraries

The development platform used was the Linux operating system. The Fedora Core 3 distribution was selected and the ANJUTA integrated development environment was used. The programming language selected was the C programming language.

In order to simplify the development of the system two open source programming libraries were adopted: the libSRTP, and the Libgcrypt. The first is an implementation[11] of the Secure RTP [45] implemented by Cisco. The second is a general purpose cryptographic library[12] that provides cryptographic functions for symmetric ciphers (AES, DES, Blowfish, CAST5, Twofish, Arcfour), hash algorithms (MD4, MD5, RIPE-MD160, SHA-1, TIGER-192), message authentication codes (HMAC for all hash algorithms), public key algorithms (RSA, ElGamal, DSA), large integer functions, and random numbers.

[11] libSRTP is available at the http://libsrtp.sourceforce.net.

[12] Libgcrypt is available at http://directory.fsf.org/security/libgcrypt.html.

5.4.3. Encryption algorithms

The programming language adopted enables the use of a variety of cryptographic algorithms. The selection of the algorithms to use was based on public acceptation, security and performance. RSA[13] was selected as the asymmetric encryption algorithm, and the Rijndael algorithm, or AES[14], as the symmetric encryption algorithm.

RSA [46] is a widely used public-key algorithm suited for signing and encrypting. It is considered secure provided long keys are used. Its strength resides in factoring very large numbers. Considering the existence of a fast factorization method, it might be possible to break RSA. The largest number factored by general-purpose methods was 576 bits long, on 3 of December of 2003[15].

The Rijndael algorithm [47] was adopted in 2001 by the National Institute of Standards and Technology (NIST) of the United States as the Advanced Encryption Standard (NIST FIPS PUB 197). It should substitute the Data Encryption Standard or DES algorithm (NIST FIPS 46-3).

5.5. The innovation of the system

We claim that the MD is a new network element. Its new properties include the capability of optimizing last mile network usage according to Receivers request, and on filtering undemanded video channel transmissions. It enables the control of individual Receiver access to the video channels transmitted in the last mile area network through the use of secure multicast. It is scalable, and offers fast response times in video channel zapping situations.

The solution proposed is based on the architecture proposed by the IETF's MSEC working group. This influence can be observed specially in the decentralized group management by splitting it into several subgroups, each one having a local group manager and a key distributor. The major difference between these two architectures is on the data transmission path. In our solution the MD is part of the data path; in MSEC's this does not happen. The solution proposed requires traffic filtering at the edge of the core network, whereas the MSEC architecture does not consider the data path, which is assumed to be a multicast transmission.

---

[13] RSA are the initials of the inventors of the algorithm (Rivest, Shamir and Adleman).

[14] Advanced Encryption Standard.

[15] By J. Franke (RSA Factoring Challenge - http://www.rsasecurity.com/)

Another difference is the importance given to source authentication. In MSEC this is required and supplied; in our solution it is not considered a key issue.

A comparison with traditional cable TV services can be done because both systems enable the distribution of simultaneous video channels and have access control mechanisms. Cable TV uses FDM and analog video transmission. Our solution is all-IP. The access control in cable TV is related to the physical access to the transmission medium, to which derivations may be made and be undetected. In our solution this is not possible since the contents of every channel are encrypted.

6. Evaluation of the secure video distribution system

This chapter focuses on the evaluation of our solution, made by a set of tests whose results are presented. Aspects such as the description of the testbed, the test methodology and the discussion of the results are addressed.

The methodology adopted for evaluating our system is based on passive testing; the system is used according well-known scenarios and traffic and signaling messages are captured using appropriate tools. This methodology becomes useful both in functional and performance tests, since it enables packet observation with arrival time, and verification of the packet structure. In addition, using the traffic capture application, we could even add new functions which were used for additional debugging. The network traffic capture application used was the Ethereal[16], for which a dissector was developed. The latter was used to analyze the network protocols developed (IMGP, KDP), as well as the other packets exchanged between the MD modules.

The chapter starts by presenting the functional tests used to verify that the requirements identified in Chapter 5 are met. For that purpose, we present some captures of live network traffic. The performance tests follow, and the aim here is to demonstrate the behavior of the protocols proposed and other packets exchanged from a timed perspective. At the end of the chapter we give an overview of the dissector we have developed and that was used to parse the packets captured.

---

[16] Available at http://www.ethereal.com.

6.1. Functional tests

These functional tests aim to demonstrate that the requirements identified are met.

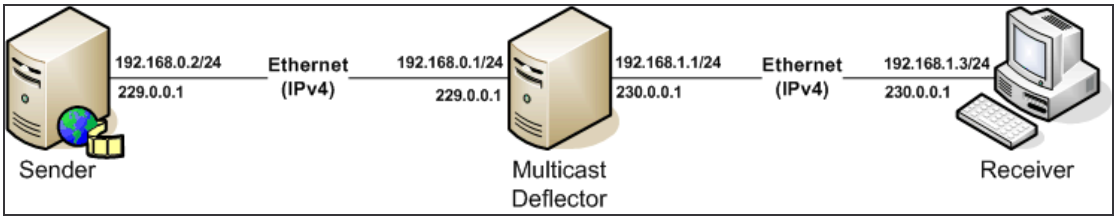

Figure 37 – Testbed #1

The testbed used consists of 3 computers, each one representing a system element; they are Intel-based architectures (Table 11), and run Fedora Core 3 Linux. IPv4 was used over 100 Mbps full-duplex Ethernet, which fits well to the reference scenario, particularly for the last mile network area where a broadcast layer 2 network is expected. Each Ethernet segment has its own IPv4 subnet. The first represents the core network and the second represents a last mile local area. The network segments are interconnected by the MD, that represents the edge between the core and the last mile networks, through its 2 Ethernet network interfaces.

	Sender	MD	**Receiver**
CPU	AMD Athlon 64 Processor 3000+	AMD Athlon 64 Processor 3000+	AMD Duron Processor 600 Mhz
RAM	512Mb	512Mb	384Mb
HD	80Gb	80Gb	20Gb
NIC	100Base-TX	100Base-TX	100Base-TX

Table 11 – Testbed #1 elements hardware characteristics

Requirement 1 says that encrypted data transmission must be used, in order to prevent unauthorized content access or disclosure. The data transmission refers to the video channel transmission, i.e. to the RTP channel stream. In order to encrypt the RTP stream the SRTP was used. The differences between them are small because the RTP and the SRTP headers have the same structure, making the distinction between RTP and SRTP packets a difficult task. The traffic capture application identifies both types of packets as RTP traffic (Figures 38 and 39), parses correctly their headers, and identifies correctly the payload type, the synchronization source, the timestamp, and the packet sequence number. The latter is the field that enables us to distinguish between RTP and SRTP, because it uniquely identifies each packet in the video stream and it is not modified when SRTP encryption is applied to the packet.

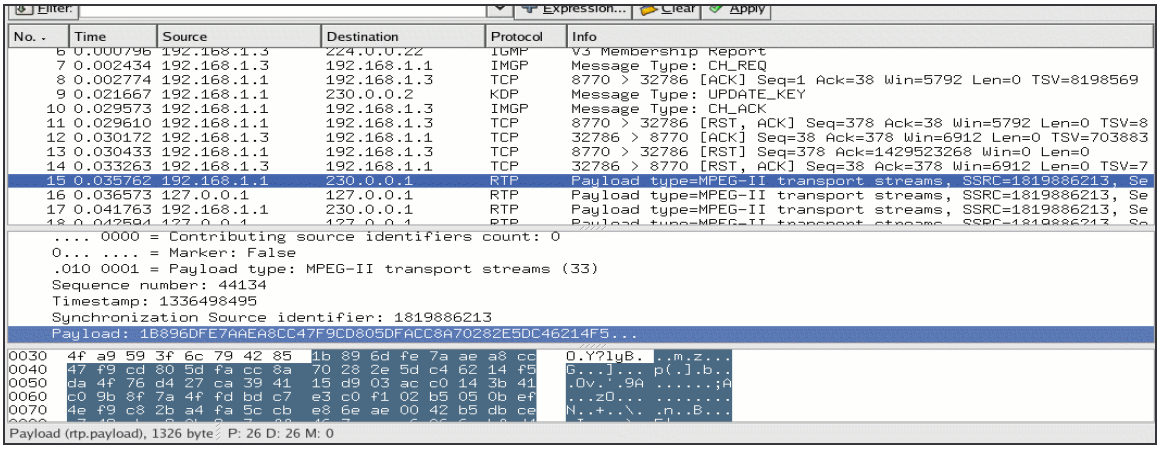

Figure 38 – SRTP packet highlight

Figures 37 and 38 show a traffic capture on the network interfaces of the Receiver element. Considering that the SRTP stream received by the Receiver is decrypted and forcing it to be forwarded to its loop back interface, we can identify the same portion of the stream by verifying that both the RTP and the SRTP packets have the same sequence number. Figure 38 highlights the SRTP packet having the sequence number 44134 that is sent in multicast by the MD. Figure 39 shows the RTP packet having the same sequence number, timestamp and synchronization source identifier, that is, the same information, but now with a different payload since it is decrypted.

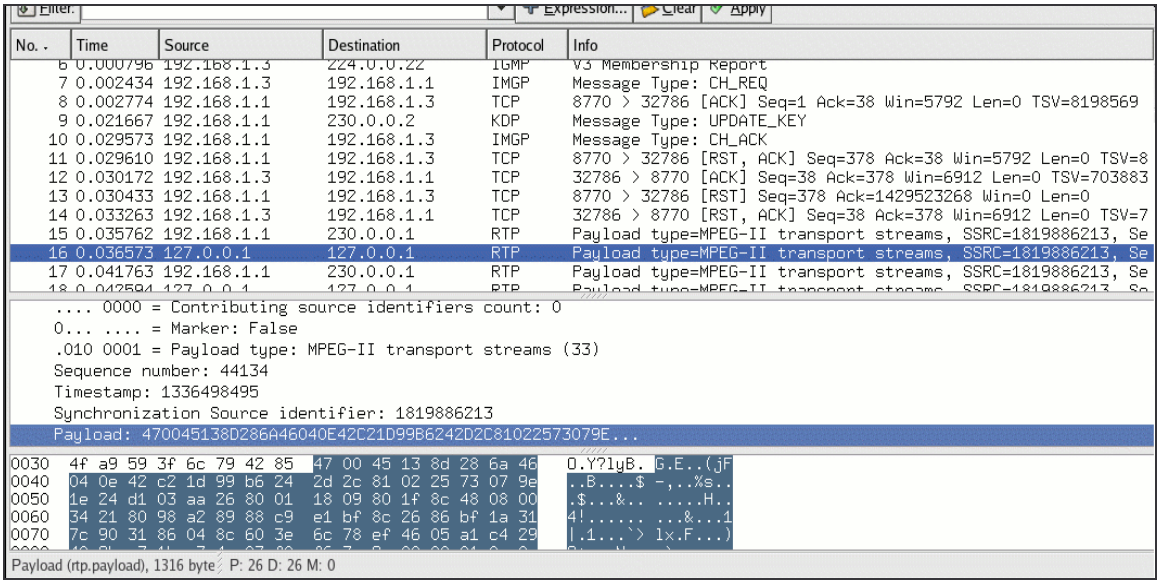

Figure 39 – RTP packet highlight

Individual and unique identification of video channel streams and Receivers is made by means of simple sequential number generation and assignment. It fulfils Requirement 3, and partially Requirement 2, which also demands a mechanism to authenticate each Receiver. Authentication is verified by the IMGP module at the MD and it is initially based in a pre-shared symmetric cryptographic key, that is used by the Receiver in the encryption of the

channel request (CH_REQ) messages. If the IMGP module at the MD can decrypt correctly the channel request, then the user is considered authenticated. Figure 40 shows a capture of an authentication failure during the channel request. This failure was generated by simply changing the pre-shared key at the Receivers side making the Receiver unable to authenticate itself. In Figure 40, the request message can be easily identified; it is highlighted and the payload starts with the string 'REQ'. After this request no other message is returned from the MD, that immediately ends the TCP connection since the request made is not considered valid.

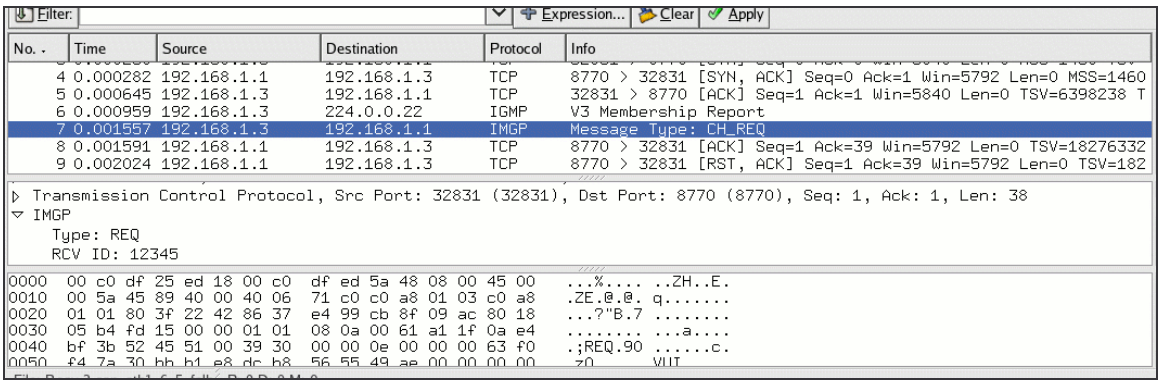

Figure 40 – IMGP CH_REQ message highlight

Requirement 4 considers access control by user and by channel. If an authenticated user requests a channel for which it does not have authorization, this request must be rejected. This is accomplished by first authenticating the Receiver and then comparing the channel requested with the list of authorized (i.e. subscribed) channels for that Receiver. If there is a match, the Receiver is authorized and the channel is streamed; if not, the request is not considered and the connection is reset. The access control by video channel is achieved by distributing the CEK to the Receivers authorized (1 copy per Receiver) and encrypted with the Receivers session key (Figure 41). The UPDATE_KEY message is used for that purpose. The channel source is authenticated through cryptographic techniques. For instance, all the messages sent by the server are digitally signed, and each video SRTP stream is encrypted so the source can be authenticated. Two techniques are used to guarantee source authentication of the SRTP stream; the first consists in the decryption of the stream by the Receivers, what means that the stream was encrypted with the correct key; the second uses the capability of the libSRTP to sign the messages. Figure 42 highlights a channel request acknowledge message (CH_ACK) sent by the MD to the Receiver to confirm the request. In order to protect the Receiver against messages sent by ill-intentioned senders, the CH_ACK message contains a field that is signed with the MD private key.

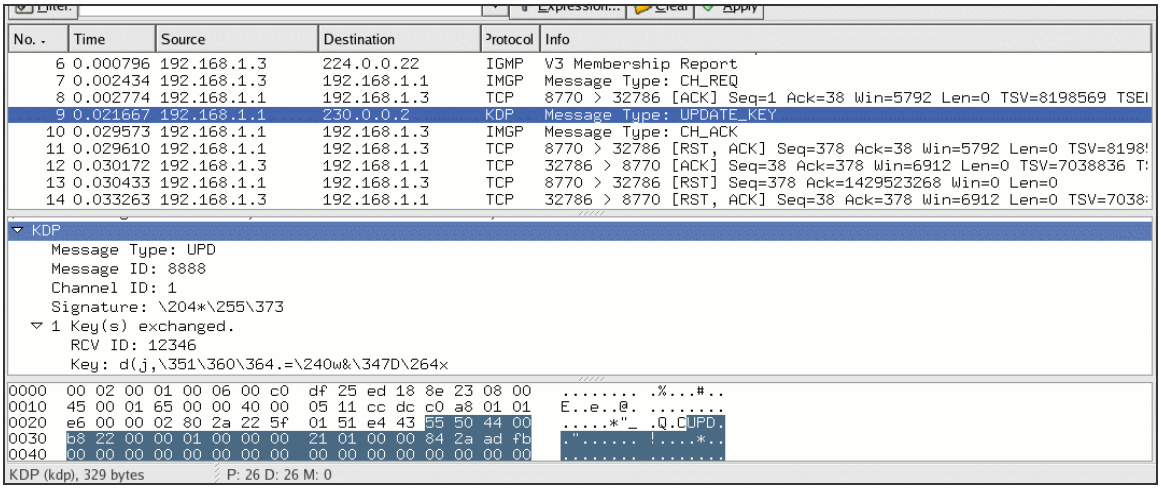

Figure 41 – IMGP UPDATE_KEY message highlight

Requirement 6 refers to the transmission of video channels, and can be identified in Figures 38, 39 and 42. The video channel streams in the last mile network are sent to the group address (multicast address) 230.0.0.1, which is the address configured for the transmissions of the encrypted channels in the last mile network. The video streams in the core network are sent with no encryption and multicast to group 229.0.0.1.

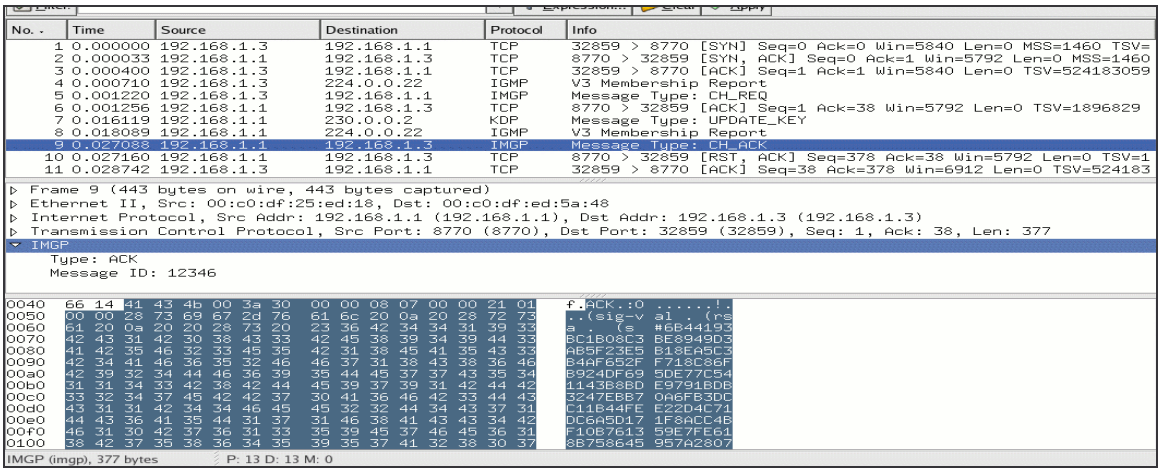

Figure 42 – IMGP CH_ACK message highlight

Requirement 7 says that channels must be transmitted in the last mile network only when they are required. This means that a video channel stream shall appear in the network capture only if there was a previous channel request (CH_REQ), which was accepted (CH_ACK) by the MD. This behavior is shown in Figure 42. Figure 40 shows a request which was rejected and, no video channel is transmitted.

The last two requirements, Requirements 8 and 9, are related with Requirement 4. The user mobility restriction is achieved through access control mechanisms, which make the authorization process fail if the credentials presented do not belong to the correct local group, that is, are not associated to the correct MD element. As a consequence, if a failure occurs in a

MD element only the Receivers assigned to that MD cease to operate, what improves the scalability of the proposed solution.

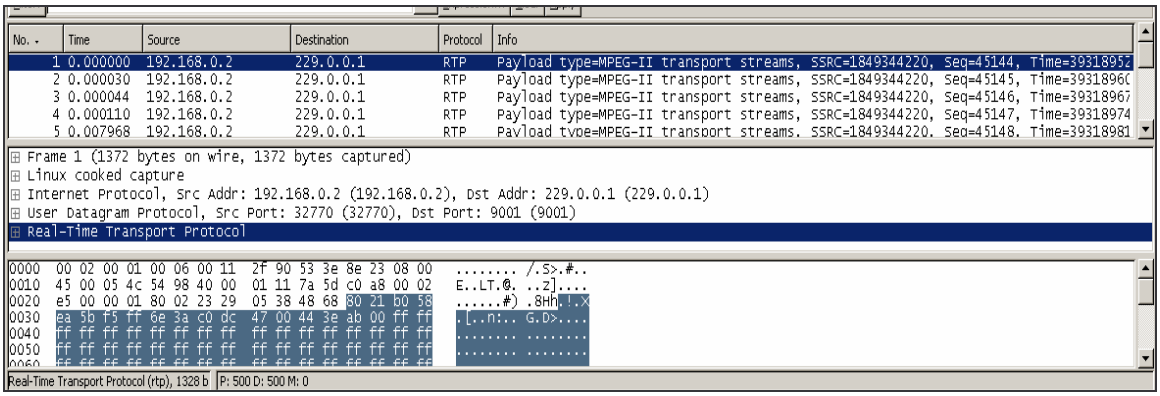

Figure 43 – Core network RTP stream highlight

6.2. Performance tests

The testbed used for performance tests is shown in Figure 44. It consists of a MD, 2 Receivers and 1 Monitor. The aim of these tests is to show the limits of the MD when processing the signaling sent by the Receivers. Receivers are on charge of generating signaling, MD will process it, and the Monitor will capture traffic and make measurements.

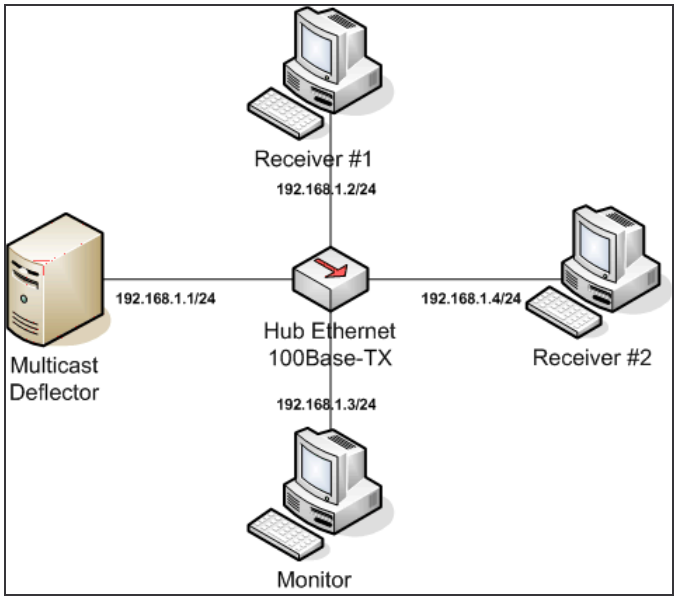

Figure 44 – Testbed #2

The main parameter to evaluate is the channel request processing time, because it significantly influences the global performance of the system. The test made to evaluate this parameter consists in sending to the MD a channel requests whose rate will increase until the MD's CPU becomes 100% used. For this reason, the values obtained depend on the MD CPU power and implies the existence of multiple Receivers.

	MD	Receiver #1	Receiver #2	Receiver #3
CPU	AMD Athlon 64 Processor 3000+	AMD Athlon 64 Processor 3000+	Intel Pentium M Processor 1400 MHz	AMD Duron Processor 600 MHz
RAM	512Mb	512Mb	512Mb	384Mb
HD	80Gb	80Gb	20Gb	20Gb
NIC	100Base-TX	100Base-TX	100Base-TX	100Base-TX

Table 12 – Testbed #2 elements hardware characteristics

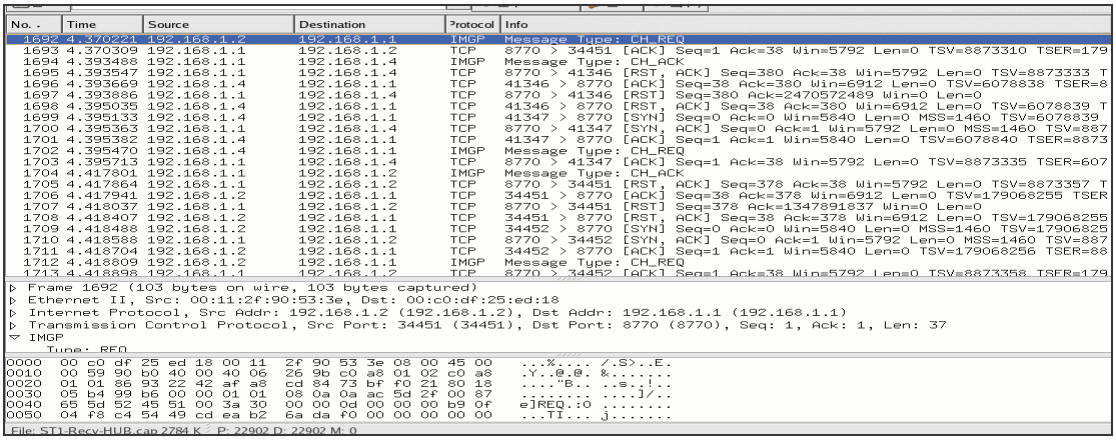

Figure 45 – Performance test packet list

The main objective of this test is to calculate the average rate of channel requests that the MD is able to process and the bandwidth needed for this number. In order to carry out the test the Receivers code was modified. The code not required for channel request processing was removed, which lead to a cleaner network capture, as shown in Figure 45. The log contains only CH_REQ and CH_ACK messages. A summary of the performance test network capture is shown in Figure 46, and presents a capture duration of 59 seconds, with an average traffic rate of 0,34 Mbit/s.

Traffic	Captured	Displayed
Between first and last packet	58,844 sec	
Packets	22902	
Avg. packets/sec	389,200	
Avg. packet size	108,501 bytes	
Bytes	2484889	
Avg. bytes/sec	42228,560	
Avg. MBit/sec	0,338	

Figure 46 – Performance test summary

During this 59 second capture, 2291 TCP/IP connections were identified between the Receivers and the MD (Figure 47), meaning that 2291 channel request were processed in that time interval, that is approximately 39 video channel requests were processed per second by

the MD. During the processing of the video channel requests, the MD reached a CPU utilization of 100%.

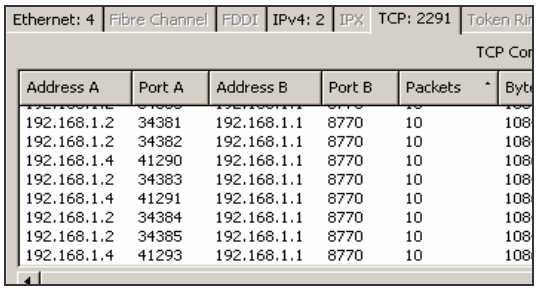

Figure 47 – Performance test TCP/IP conversations

6.3. Ethereal dissector implementation

The Monitor tool used to evaluate our systems is based on the Ethereal tool [48]. Ethereal includes a protocol analyzer that is able to recognize many protocols and enables the addition of new protocols through the development of Ethereal Protocol Dissectors. The protocol dissectors are the portions of code used by the Ethereal to analyze (parse and display) the packets captured for a protocol. They are coded in C language.

```
1.     void
2.     proto_register_imgp(void)
3.     {
4.            /* Section 1 - Protocol registration */
5.            proto_imgp = proto_register_protocol("IMGP", "IMGP","imgp");
6.            /* Section 2 - Header fields registration*/
7.            static hf_register_info hf[] = {
8.                   { &hf_imgp_command,
9.                   { "Type", "imgp.command",FT_STRING,
10.                         BASE_NONE, NULL, 0x0, "Message type", HFILL }},
11.
12.                  { &hf_imgp_rcvID, {"RCV ID", "imgp.rcvID", FT_INT32,
13.                         BASE_DEC, NULL, 0x0, "Receive ID.", HFILL}},
14.
15.                  { &hf_imgp_msgID, {"Message ID", "imgp.rcvID", FT_INT32,
16.                         BASE_DEC, NULL, 0x0, "ID of the message.", HFILL}}

17.           };
18.           proto_register_field_array(proto_imgp, hf, array_length(hf));
19.           /* Section 3 - Protocol subtree registration*/
20.           static gint *ett[] = {
21.                  &ett_imgp,
22.           };
23.           proto_register_subtree_array(ett, array_length(ett));
24.    }
```

Table 13 – Excerpt of the dissector for the IMGP protocol - the *proto_register_imgp* function

The layered nature of network traffic imply that dissectors run on top of other, lower layer, protocol dissectors. The IMGP dissector, for instance, runs after the TCP dissector. The dissector development, for a protocol using TCP, consists in the addition of three main functions: 1) the registration of protocol and the variables in Ethereal; 2) the registration of the current protocol as a sub-dissector of another protocol; and 3) the protocol dissection

function. The first function is used to inform the Ethereal about the packets attributes that will be present on the screen and about the data structures, such as packet header fields (Table 13, line 8), protocol visualization trees and sub-trees (Table 13, line 22) and the protocol itself (Table 13, line 5).

```
1.     void proto_reg_handoff_imgp(void)
2.     {
3.            dissector_handle_t imgp_handle;
4.            imgp_handle = create_dissector_handle(dissect_imgp,
proto_imgp);
5.            dissector_add("tcp.port",IMGP_PORT,imgp_handle);
6.     }
```

Table 14 – Excerpt of the dissector for the IMGP protocol - the *proto_reg_handoff_imgp* function

The second function, registration of the dissector as a sub-dissector of another protocol, uses a pointer to the start of the packet that is considered by the dissector as payload or data. The code presented in Table 14 shows a sub-dissector registration process, which uses a dissector handle that will be used by Ethereal to identify and call the current dissector and the sub-dissector. In this example, the IMGP protocol dissector is registered as a sub-dissector of the TCP protocol and it is associated with the TCP/IP destination port IMGP_PORT, as shown in Table 14, line 6.

```
1.     static void
2.     dissect_imgp(tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree)
3.     {
4.            proto_item *ti;
5.            proto_tree *imgp_tree;
6.
7.            if(check_col(pinfo->cinfo,COL_PROTOCOL))
8.                   col_set_str(pinfo->cinfo, COL_PROTOCOL,"IMGP");
9.
10.           if(check_col(pinfo->cinfo,COL_INFO)){
11.                  if(isMessageType(tvb,"REQ"))col_set_str(pinfo->cinfo,
COL_INFO, "Message Type: CH_REQ");
12.                  if(isMessageType(tvb,"ACK"))col_set_str(pinfo->cinfo,
COL_INFO, "Message Type: CH_ACK");
13.                  if(isMessageType(tvb,"NAK"))col_set_str(pinfo->cinfo,
COL_INFO, "Message Type: CH_NAK");
14.           }
15.
16.           if(tree){
17.                  ti = proto_tree_add_item(tree,proto_imgp,tvb,0,-
1,FALSE);
18.                  imgp_tree = proto_item_add_subtree(ti, ett_imgp);

19.
20.                  if(isMessageType(tvb,"REQ") || isMessageType(tvb,"ACK")
|| isMessageType(tvb,"NAK")){
21.
      proto_tree_add_item(imgp_tree,hf_imgp_command,tvb,0,4,FALSE);
22.
23.                        if(isMessageType(tvb,"REQ")){
24.
      proto_tree_add_item(imgp_tree,hf_imgp_rcvID,tvb,4,4,TRUE);
25.                        }
26.
27.                        if(isMessageType(tvb,"ACK") ||
isMessageType(tvb,"NAK"))
28.                        {
```

```
29.
      proto_tree_add_item(imgp_tree,hf_imgp_msgID,tvb,4,4,TRUE);
30.                         }
31.                 }
32.           }
33.     }
```

Table 15 – Excerpt of the dissector for the IMGP protocol - the *dissect_imgp* function

The last function, the dissection, is executed when Ethereal processes a packet that is associated with a TCP connection and addressed to the IMGP port number. Each call to the dissection function of the IMGP dissector means that a new TCP payload arrived and it needs to be dissected by the protocol dissection function (Table 15). Protocol dissection functions are prototyped to accept three arguments: *tvb*, *pinfo* and *tree*. The *tvb* argument is a pointer to a data structure of type *tvbuff_t* and points, in our case, to the TCP packet payload. The *pinfo* argument is a pointer to a data structure holding additional packet information which is gathered by lower layer dissectors; it is used typically to manipulate the packet list pane (Figure 48) and it writes text into the info and protocol columns for each packet dissected (Table 15, line 7-14). The *tree* argument is a pointer to a data structure that allows the manipulation of packet details pane; these details are represented as a tree consisting of sub-trees of items, each sub-tree corresponding to a dissector. In the example of Table 15, the line 20 shows the addition of an item to the IMGP protocol tree that corresponds to the first four bytes of the payload, that is, the message type.

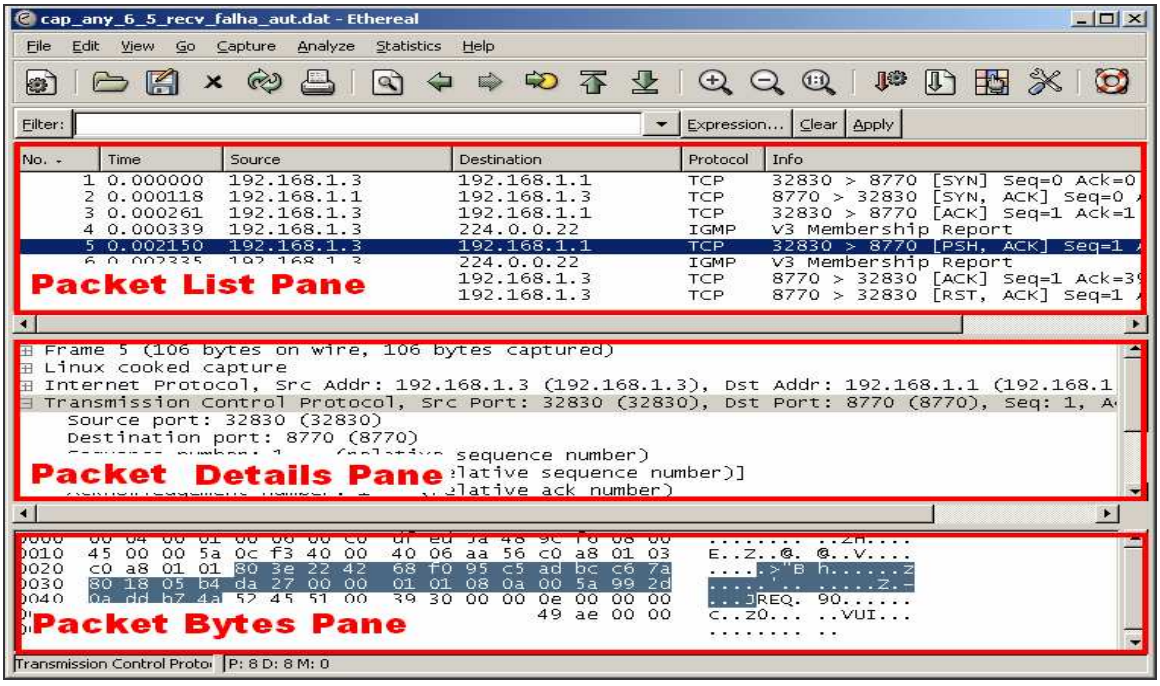

Figure 48 – Ethereal panes

7. Conclusion

7.1. Revision of the work

The first chapter of this thesis presented its main objectives. They consisted on the definition of a system capable of generating and distributing multiple video channels over a secure IP multicast infrastructure. This system should grant the access of individual users to individual video channels, and on the optimization of the network usage in the *last mile* network, so only the channels required by the clients are transmitted. The reference scenario consists of three elements: video server, video distribution system, and client. The video server is responsible for generating a video channel and streaming it as a sequence of IP packets. The video distribution system is responsible for distributing the CEKs, processing client requests, authenticating and authorizing clients, and for retransmitting the video channels requested. The client interacts with the key distribution system in order to request video channels, authenticate itself, obtain CEKs, and obtain, decrypt and display the video channels sent as IP packets.

One of the assumptions of the work presented in this thesis is the transmission of IP packets to a group, i.e., IP multicast transmission. This concept takes advantage of the functions provided by the IP protocol to create and manage groups of network elements, which enable a single packet to be received by a group. In Chapter 2 we presented the state-of-the-art in multicast transmission in IPv4 and IPv6 networks, described their group management protocols (IMGP and MLD), presented some routing protocols, and introduced layer 2 multicast. An overview of current efforts on multicast transmission in future 3rd generation

networks for multimedia content transmission, and the adaptation of multicast to satellite transmissions were also given.

Video streaming over IP is defined in the IETF as a multimedia architecture which includes protocols such as RTP. In Chapter 3 we described the IETF multimedia architecture. In the data plane we addressed RTP, suited for end-to-end transmission of real-time data without QoS guaranties, and its companion protocol RTCP which enables receivers to provide information about the QoS to the sender. In the control plane, the IETF suggests the adoption of SIP or RTSP signaling protocols; both allow the initiation and the termination of streaming sessions. SIP supports user mobility through the use of proxies, while RSTP is aimed at simulating a VCR-like session control.

Chapter 4 focused on network security. Cryptographic techniques such as symmetric encryption, asymmetric encryption, and secure group communication were presented. The current IETF efforts in this area, made by the MSEC working group, were presented in detail. Despite the MSEC efforts, no optimal solution has yet been defined. Works carried out by other researchers were also described and discussed.

Chapter 5 presented our solution. Some scenarios were identified and based on them, we identified 9 requirements: 1) encrypted data transmission – video channel streams must be encrypted prior to their transmission; 2) unique user identification and authentication – each user must be identified individually in order to authenticate itself and enabling access control; 3) unique channel identification – it must be possible to distinguish each channel from the others; 4) access control by user and channel – it must be possible to prevent an authenticated user from accessing channels for which it has no authorization; 5) channel source authentication – it must be possible for a client to be sure that it is receiving data from the correct server; 6) group transmission – each video channel should be transmitted only once for all the interested clients; 7) transmit only if required – if there are no clients interested in a video channel it shall not be transmitted; 8) restringe user mobility – a set of user credentials should only be valid in the user's network area; 9) scalability – is should be possible to support many users as required. The system design followed. The architecture of the system proposed is heavily based on the MSEC reference framework and consists of 3 elements: the MD, the Sender and the Receiver. Our Receiver element is analogous to the Receiver element of the MSEC framework; our Sender element is similar to the MSEC Sender. The group key management functional area of the MSEC framework is also implemented in our MD element, having the same responsibilities of the MSEC's GCKS, which include user authentication, user authorization, key distribution and generation. It also has functions of the MSEC's Policy Server such as GSAs or rekey interval definition. New protocols were

developed and implemented. IMGP is a simple protocol which manages groups of users interested in a channel, and accepts user channel requests. In IMGP, the group join operation is represented by the CH_REQ message, that may be confirmed through an acknowledge message (CH_ACK) or not (CH_NAK); the group leave operation is implicit, that is, based on timeouts and, therefore, using no protocol messages. The KDP allows the distribution of CEK to authorized receivers, and it is based in the key distribution messages adopted in Iolus framework [36] and on the protocol described in [43]. At the end of the chapter, the innovation of the proposed solution was characterized, which include the capability of optimizing last mile network usage according to Receivers requests, and on filtering undemanded video channel transmissions. Therefore, the proposed system enables the control of individual Receiver access to the video channels transmitted in the last mile area network through the use of secure multicast. It is scalable and offers fast response times in video channel zapping situations.

In Chapter 6 we evaluated the implementation of the proposed solution through both functional and performance tests over the secure video distribution system. The methodology adopted for evaluating the system is based on passive testing; the system is used according to well-known scenarios, and traffic and signaling messages were captured using appropriate tools. This is useful in both cases, since it enables packet observation with arrival time, and verification of the packet structure. The performance tests showed the limits of the MD when processing the signaling sent by the Receivers, in order to calculate the average rate of channel requests that the MD is able to process and the bandwidth required. Channel requests were sent to the MD, at rates which increased until the MD's CPU became 100% used. During the performance testing, 2291 TCP/IP connections were identified between the Receivers and the MD over a period of 59 seconds, meaning that 2291 channel request were processed in that time interval, that is, 39 video channel requests processed per second by the MD. The functional tests demonstrated that the requirements identified during the system specification are met. Requirement 1 says that encrypted data transmission must be used; so, in order to encrypt the RTP stream, SRTP was used and the same payload was identified in both forms through the sequence number field of the packets. Unique identification of video channel streams and Receivers is made by means of sequential number generation and assignment, what fulfills Requirement 3 and partially fulfills Requirement 2; Requirement 2 also demands a mechanism to authenticate Receivers, what is done by the IMGP module at the MD and based on an initial pre-shared symmetric cryptographic key. Requirement 4 considers access control by user and by channel and it is accomplished by first authenticating the Receiver and then comparing the channel requested with the list of authorized (i.e. subscribed) channels for that Receiver. Requirement 6 refers to group transmission and was done by streaming channels to multicast addresses. Requirement 7 says that channels must be

transmitted in the last mile network only when they are required; this was achieved through the MDE module at the MD, that signals the starting and the stopping of channel streams. Requirement 8 states that user mobility should be limited, and Requirement 9 states that the system must by scalable; user mobility restriction is achieved through access control mechanisms, which make the authorization process fail if the credentials presented do not belong to the correct local group, that is, are not associated to the correct MD element. As a consequence, if a failure occurs in a MD element only the Receivers assigned to that MD cease to operate, what improves the scalability of the proposed solution.

7.2. Characterization of the results

The work carried out in this master thesis lead us to four main results: 1) video transmission system architecture; 2) multicast filtering network element; 3) group management and access control mechanism; and 4) video system prototype. To the best of our knowledge the second result – the multicast filtering network element – is new and it constitutes a contribution.

The video transmission system architecture supports a video transmission system capable of transmitting simultaneous video channels over secure IP multicast. The video channels are encrypted using symmetric encryption but maintaining access control per user and per channel. This architecture is composed of four modules: MDE, SRTP, IMGP, and KDP. The MDE is the controller module, and it generates new cryptographic keys (SEK or CEK) upon requests. The SRTP module receives streams sent by the Sender, either RTP or SRTP, and encrypts them with the corresponding CEK into a new SRTP stream, which is sent to a network area associated to a multicast address. The IMGP module is responsible for Receiver authentication and SEK generation at the time of reception of the channel request. The KDP module implements the KDP that allows CEK distribution to all SEK owners.

The multicast filtering network element transmits to the last mile network only the video channel streams requested by clients. It is achieved through a combination of multicast transmission, multicast routing tables, and management of client requests. The MD receives all the video channels, either in RTP or in SRTP. It has pre-configured a multicast routing table that forces the transmission of the channel to a multicast address through the last mile network interface card. These multicast addresses are those expected by the Receivers and they are associated to areas. The video channel sent by the MD can only be triggered by an authorized Receiver, through the use of the IMGP protocol.

The group management and the access control mechanism enables the formation of groups interested in the same data, but still controlling the access of individual clients to the data. The IMGP and the KDP protocols are used for that purpose. IMGP is a two message protocol that

enables clients to join groups; it assumes the client leaves after a period of inactivity. The IMGP also supports client authentication to prevent abuse or misusage, through conventional encryption and pre-sharing of cryptographic keys. It forces each user to have an individual key that will be used to obtain each SEK, that is also unique. The KDP protocol uses unique SEK to create a new CEK distribution message, and concatenates multiple CEKs, one for each interested Receiver, encrypted with these Receivers SEKs.

The video system prototype consists in a 3 PC test bed, each one with different functionalities. One acts as a video server, another as the video distributing system (MD), and the third emulates a Receiver. All of them are Intel-based architectures running Fedora Core 3 Linux. The network consists of IPv4 over 100 Mbits full-duplex Ethernet. Each Ethernet network segment is an IPv4 subnet. The first represents the core network, and the second represents the last mile network. The MD interconnects both networks.

7.3. Future work

Future work opportunities arose during the development of this thesis. They can be described from two perspectives: improvement of the current scenario, and adaptation of the current solution to different scenarios.

Several improvements can be made: 1) use only one instance of the IMGP module at the MD per Receiver; 2) obtain information about the users profile; 3) include support for Authentication, Authorization and Accounting (AAA); 4) support for pre-paid user access. The first can be achieved by maintaining open the unicast TCP connection between the Receiver and the MD. The second can be obtained by logging successful video channel requests and the end of their transmissions. The third can be achieved be adapting the IMGP module so that it starts to validate Receiver credentials in an AAA server; it would also enable accounting possibilities. The last can also be achieved using the IMGP module but, now, interacting with a credit control server. The user profile characterization could lead to the optimization of core network, by allowing the MD to not subscribe the multicast groups associated to the channels which are never viewed by its local Receivers.

The adaptation of the solution proposed to different types of last mile media, simultaneously or not, arises also as interesting, particularly when considering group management and key distribution. An example could be the adaptation of the solution to non-broadcast media having bandwidth limits, such as ADSL. This work, in particular, is being carried out under a contract with a Portuguese telecom operator.

References

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," in *RFC 3550*, 2003.

[2] A. Pinto and M. Ricardo, "Multiple Video Channel Transmission using Secure IP Multicast," presented at Conftele2005 - 5th Conference on Telecommunications, Tomar, 2005.

[3] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3," in *RFC 3376*, 2002.

[4] S. Deering, W. Fenner, and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6," in *RFC 2710*, 1999.

[5] R. Vida and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6," in *RFC 3810*, 2004.

[6] A. Conta and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," in *RFC 2463*, 1998.

[7] G. Malkin, "RIP Version 2," in *RFC 2453*, 1998.

[8] J. Moy, "OSPF Version 2," in *RFC 2178*, 1998.

[9] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," in *RFC 1075*, 1998.

[10] J. Moy, "Multicast Extensions to OSPF," in *RFC 1584*, 1994.

[11] J. Moy, "MOSPF: Analysis and Experience," in *RFC 1585*, 1994.

[12] A. Adams, J. Nicholas, and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)," in *RFC 3973*, 2005.

[13] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," in *RFC 2362*, 1998.

[14] 3GPP, "Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS); Architecture and functional description (Release 6)," in *TS 23.246 V.6.5.0*, 2004.

[15] S. Praveenkumar, H. Kalva, and B. Furht, "Application of Video Error Resilience Techniques for Mobile Broadcast Multicast Services (MBMS)," presented at Sixth International Symposium on Multimedia Software Engineering (ISMSE'04), 2004.

[16] ETSI, "Digital Video Broadcasting (DVB); DVB specification for data broadcasting," in *EN 301 192 V1.4.1*, 2004.

[17] Z. Sun, M. Howarth, H. Cruickshank, S. Iyengar, and L. Claverotte, "Networking issues in IP multicast over satellite," *International Journal of Satellite Communications and Networking*, pp. 489-507, 2003.

[18] D. Wu, Y. Hou, W. Zhu, Y. Zhang, and J. Peha, "Streaming Video over the Internet: Approaches and Directions," *IEEE Transactions On Circuits and Systems for Video Technology*, vol. 11, pp. 282-300, 2001.

[19] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session initiation protocol," in *RFC 2543*, 1999.

[20] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," in *RFC 2326*, 1998.

[21] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol - HTTP/1.1," in *RFC 2616*, 1999.

[22] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Third ed: Prentice Hall, 2003.

[23] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: A New Approach to Group key Agreement," presented at IEEE ICDCS'98, 1998.

[24] M. Baugher, R. Canetti, L. Dondeti, and F. Lindholm, "MSEC Group Key Management Architecture," in *Internet Draft, draft-ietf-msec-gkmarch-08.txt*, 2004.

[25] T. Hardjono, R. Canetti, M. Baugher, and P. Dinsmore, "Secure IP Multicast: Problem Areas, Framework, and Building Blocks," in *Internet Draft, work in progress*, 2000.

[26] A. Perrig, R. Canetti, B. Briscoe, D. Tygar, and D. Song, "TESLA: Multicast Source Authentication Transform Introduction," in *Internet Draft, draft-ietf-msec-tesla-intro-04.txt*, 2005.

[27] M. Baugher, B. Weis, T. Hardjono, and H.Harney, "The Group Domain of Interpretation," in *RFC 3547*, 2003.

[28] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman, "MIKEY: Multimedia Internet KEYing," in *RFC 3830*, 2004.

[29] H. Harney, U. Meth, A. Colegrove, and G. Gross, "GSAKMP: Group Secure Association Group Management Protocol," in *Internet Draft, draft-ietf-msec-gsakmp-sec-10.txt*, 2005.

[30] A. Colegrove and H. Harney, "Group Policy Token V1 with Application to GSAKMP," in *Internet Draft, draft-ietf-msec-policy-token-sec-02.txt*, 2005.

[31] B. Haberman and R. Worzella, "IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol," in *RFC 3019*, 2001.

[32] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture," in *RFC 2094*, 1997.

[33] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification," in *RFC 2093*, 1997.

[34] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, vol. 35, pp. 309-329, 2003.

[35] A. Ballardie, "Scalable Multicast Key Distribution," in *RFC 1949*, 1996.

[36] S. Mittra, "Iolus: a framework for scalable secure multicast," presented at ACM SIGCOMM '97, Cannes, France, 1997.

[37] L. Dondeti, S. Mukherjee, and A. Samal, "Scalable secure one-to-many group communication using dual encryption," *Computer Communications Journal*, pp. 1681-1701, 2000.

[38] B. Briscoe, "MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences," presented at First International COST264 Workshop on Networked Group Communication, 1999.

[39] R. Molva and A. Pannetrat, "Scalable multicast security in dynamic groups," presented at 6th ACM Conference on Computer and Communications Security, New York, 1999.

[40] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A Scalable Group Re-Keying Approach for Secure Multicast," presented at 2000 IEEE Symposium on Security and Privacy (S&P 2000), Berkeley, CA, 2000.

[41] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure group communications for wireless networks," presented at Military Communications Conference (MILCOM 2001), Reading, MA, 2001.

[42] S. Rafaeli and D. Hutchison, "Hydra: A decentralized group key management," presented at Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02), Los Alamitos, California, 2002.

[43] H. Chu, L. Qiao, K. Nahrstedt, and H. Wang, "A Secure Multicast Protocol with Copyright Protection," *ACM Computer Communication Review Journal (ACM CCR)*, vol. XXXII, pp. 42-60, 2002.

[44] H. Bettahar, A. Bouabdallah, and Y. Challal, "AKMP: An Adaptive Key Management Protocol for secure multicast," presented at Eleventh International Conference on Computer Communications and Networks, 2002.

[45] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," in *RFC 3711*, 2004.

[46] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," in *Communications of the ACM*, vol. 21, 1978, pp. 120-126.

[47] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," in *NIST AES Proposal*, 1998.

[48] G. Combs, "Ethereal: A Network Protocol Analyzer," 2005.