Universidade do Porto

Faculdade de Engenharia

FEUP

Fernando Joaquim Lopes Moreira

Gestão de Recursos e Controlo De Tráfego em Redes com Integração de Serviços

Porto, Abril 2003

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Departamento de Engenharia Electrotécnica e de Computadores

Gestão de Recursos e Controlo de Tráfego em Redes com Integração de Serviços

Fernando Joaquim Lopes Moreira

UNIVERSIDADE DO PORTO
BIBLIOTECA

Licenciado em Informática/Matemáticas Aplicadas

Mestre em Engenharia Electrotécnica e de Computadores

(Área de Telecomunicações)

Porto, Abril 2003

004(043)/MOR f/GES
UNIVERSIDADE DO PORTO
Faculdade de Engenharia
BIBLIOTECA M
N.º 68515
CDU 004(043)
2 / 10 / 2003

Tese submetida à Faculdade de Engenharia da Universidade do Porto

para a obtenção do grau de Doutor

em

Engenharia Electrotécnica e de Computadores

(Area de Telecomunicações)

Tese realizada sob a supervisão de

Doutor José António Ruela Simões Fernandes, Professor Associado

do Departamento de Engenharia Electrotécnica e de Computadores

da Faculdade de Engenharia da Universidade do Porto

Ao Bernardo, à Maria Helena, à Elsa, aos meus pais e ao meu irmão.

Resumo

A tradicional separação das redes de telecomunicações com base nos serviços que disponibilizam tem vindo a ser substituída pelo conceito de integração de serviços numa única rede, o que exige mecanismos que garantam qualidade de serviço (QoS) diferenciada de acordo com os requisitos das aplicações. Neste contexto a necessidade de suportar elevados volumes de tráfego e requisitos de QoS muito exigentes requer a implantação de *backbones* de alta velocidade e baixa latência. A tecnologia ATM responde a estes requisitos pelo que constituiu uma primeira escolha; por outro lado a expansão/consolidação das redes IP teve como consequência natural a exploração de arquitecturas baseadas em IP e ATM.

Estas arquitecturas baseiam-se na estruturação de *backbones* numa rede de periferia (*edge*) com funções de nível 3 (tipicamente IP e protocolos de encaminhamento associados) e uma rede de núcleo (*core*) com funções de nível 2 (comutação rápida baseada em etiquetas). No entanto as soluções iniciais de integração IP/ATM baseadas em simples modelos de redes sobrepostas (*overlay*) apresentam limitações óbvias. Por outro lado, e mais recentemente foram desenvolvidos modelos de QoS em redes IP (*IntServ* e *DiffServ*). Estes dois factos influenciaram fortemente a procura de novas soluções arquitectónicas (por exemplo o MPLS). O modelo de estruturação, no entanto mantém-se, sendo a tendência actual o tratamento de tráfego IP na periferia (classificar, escalonar e agrupar) e a utilização de tecnologias de comutação rápida no *backbone* (ATM nativo ou ATM como tecnologia de comutação do MPLS), com diferentes níveis de integração das funções de encaminhamento (*routing*) e de comutação (*forwarding/switching*).

O trabalho apresentado na tese baseia-se no modelo descrito anteriormente e considera classes de tráfego IP modelados por *Token Buckets*. Este tráfego é escalonado e agregado, segundo um modelo proposto, nos dispositivos periféricos (*Edge Devices*, ED)

tendo em atenção os recursos disponibilizados pela rede. Neste contexto foi analisada e escolhida a categoria de serviço ABR devido às suas características para transportar este tipo de tráfego. Tendo em atenção esta opção na tese são discutidas e apresentadas propostas para questões relacionadas com esta categoria de serviço, tais como a distribuição da largura de banda dentro do ED de acordo com um critério de atribuição de pesos aos VCs (VCs com débitos mínimos garantidos atribuídos e outros nulos) e a consolidação de células de gestão de recursos na rede de núcleo.

Os resultados obtidos, através da realização de um conjunto de testes de simulação sobre várias configurações de referência, mostraram que o algoritmo de escalonamento é eficiente na distribuição da largura de banda, na distribuição dos atrasos e no controlo da ocupação dos *buffers*. O algoritmo de distribuição da largura de banda, baseado no critério *max-min fairness* proposto, pelos VCs ABR nos ED é eficiente na sua distribuição, mantém os *buffers* com uma ocupação reduzida e converge rapidamente para um estado estável. O algoritmo de consolidação de células de gestão de recursos e de distribuição da largura de banda consegue os mesmos índices de desempenho dos algoritmos anteriores.

Abstract

The traditional division of telecommunications networks on the basis of the services they provide has been recently replaced by the concept of service integration in a single network, which requires mechanisms that guarantee differentiated Quality of Service (QoS) according to the application requirements. In this context, the need to support both high volumes of traffic and very demanding QoS requirements requires the implementation of high speed and low latency backbones. On one hand, the ATM technology satisfies these requirements and was therefore selected in the first place. On the other hand, the expansion and consolidation of the IP networks brought as a natural consequence the exploration of architectures based on IP and ATM.

These architectures are based on the structuring of backbones in a peripheral network (edge) with functions of level 3 (typically IP and associated routing protocols) and a core network with functions of level 2 (labels switching). Nevertheless, the initial solutions of IP/ATM integration based on simple models of overlaid networks present obvious limitations. Also, more recently, QoS models have been developed in IP networks (IntServ and DiffServ). Both of these aspects had a strong influence in the search for new architectonical solutions (e.g. the MPLS). However, the structuring model is preserved since the main trend is the handling of IP traffic in the periphery (classify, schedule and aggregate) and the use of switching technologies in the backbone (ATM or ATM as switching technology of the MPLS), with different levels of integration of both routing and forwarding/switching functions.

The work presented in this thesis is based on the model previously described and considers classes of IP traffic modelled by Token Buckets. According to the model proposed, this traffic is scheduled and aggregated in the Edge Devices (ED), considering the resources provided by the network. In the referred context, the ABR service category

was analysed and, due to its characteristics, subsequently selected to carry the envisaged type of traffic. This thesis, considering the option outlined, discusses and addresses issues related with this service category such as the bandwidth distribution inside the ED according to the VCs weight assignment criterion (VCs either with minimum guaranteed debits or the null value) and the consolidation of resource management (RM) cells in the core network.

The results obtained with a set of simulation tests performed on several benchmark configurations have shown that the scheduling algorithm is efficient both in bandwidth distribution and in delays distribution as well as in controlling the buffers occupation. The algorithm for bandwidth distribution among the VCs ABR inside the ED and based on the proposed max-min fairness criterion, is efficient in its distribution, keeps the buffers with a reduced occupation and converges quickly to a stable state. The algorithm for consolidation of RM cells and bandwidth distribution reaches the same performance indicators of the former algorithms.

Agradecimentos

Ao meu supervisor, Professor Doutor José Ruela, pelo apoio prestado em todas as fases deste projecto, quer em termos profissionais pela sugestão do tema e pelas diversas "discussões" que permitiram a constante evolução deste trabalho e dos meus conhecimentos, quer em termos pessoais pela ajuda nos momentos mais difíceis e de grande tensão que um trabalho desta dimensão e duração temporal provocam.

À Universidade Portucalense Infante D. Henrique na pessoa do seu Reitor, Prof. Doutor Francisco da Costa Durão, pelo suporte prestado e ao Departamento de Informática na pessoa do seu director, Prof. Doutor Jorge Reis Lima, pelo seu apoio e incentivo.

Um agradecimento especial ao David Su e Nada Golmie do NIST, pela cedência do código do simulador.

Ao Ministério da Educação, programa PRODEP Medida 5 - Acção 5.3 concurso público 3/PRODEP/98, pelo suporte concedido ao candidato através de uma bolsa de doutoramento.

Não poderia deixar de referir um conjunto de pessoas que tiveram um papel muito importante na realização deste trabalho, mas por razões diferentes. Em primeiro lugar um agradecimento ao Prof. Doutor Carlos Mendonça e Moura e ao Prof. Doutor António Pascoal pela possibilidade que me deram relativamente à discussão de algumas ideias e questões e ao seu apoio e motivação. Em segundo lugar e com um caracter mais pessoal um agradecimento muito especial à Dr^a. Cristina Paes e ao Dr. José Milheiro e Carmo pelo apoio, motivação, encorajamento e tolerância ao longo deste trabalho. Por último um agradecimento aos restantes colegas do departamento pela forma mais ou menos participativa na resolução de pequenos problemas.

Um agradecimento especial a todos os meus amigos pelo seu encorajamento e

paciência durante o tempo de realização deste trabalho.

Ao meu pai, à minha mãe e ao meu irmão pela colaboração e compreensão em todos os momentos difíceis e pelos constantes incentivos, tornando-se assim um dos principais factores para o sucesso.

Um agradecimento muito especial aos meus filhos, Bernardo e Maria Helena, os quais privei da minha companhia, atenção e carinho indespensáveis nos seus primeiros anos de vida.

À Elsa pela preciosa ajuda a tornar possível o impossível e em especial à sua inesgotável compreensão e carinho.

Fernando Joaquim Lopes Moreira

Faculdade de Engenharia da Universidade do Porto

Abril 2003

Índice

Resumo vii

Abstract ix

Agradecimentos xi

Lista de Tabelas xix

Lista de Figuras xxi

Acrónimos xxv

1 Introdução **1**

1.1 Enquadramento 1

1.2 Objectivos e principais contribuições da tese 5

1.2.1 Objectivos 5

1.2.2 Contribuições da tese 6

1.3 Organização da tese 7

2 Enquadramento geral e caracterização de problemas **9**

2.1 Modelos de Qualidade de Serviço 9

2.1.1 Caracterização e modelação de tráfego 10

2.1.2 Aspectos a considerar em modelos de QoS 11

2.1.3 Modelo de QoS em ATM 13

2.1.4 Modelos de QoS em IP 18

2.2 Evolução das arquitecturas IP/ATM 21

2.3 Tendência actual de integração 21

2.3.1 Problemas 22
2.3.2 Modelo de integração 23
2.3.3 Protocolos da rede de acesso 24
2.3.4 Escolha da categoria de serviço para a rede de núcleo 25
2.3.5 Suporte de serviços com QoS 27
2.4 Questões relacionadas com conexões multiponto da rede ATM 29
2.4.1 Ponto-a-multiponto 29
2.4.2 Multiponto-a-ponto 31
2.4.3 Multiponto-a-multiponto 31

3 Ferramentas de análise e simulação **33**
3.1 Ferramenta matemática 33
3.1.1 Operações 34
3.1.2 Funções de referência 35
3.1.3 Curvas de Chegada 36
3.1.4 Curvas de Serviço 36
3.1.5 Limites 37
3.2 Simulação 38
3.2.1 O porquê da simulação ? 38
3.2.2 Decisões a ter em consideração na realização de experiências de simulação 38
3.3 O simulador 40
3.3.1 Componentes 41
3.3.2 Formato dos resultados 43
3.3.3 Componentes e aplicações desenvolvidos 43
3.4 Cenários de simulação 45

4 Trabalho relacionado **47**
4.1 Introdução 47
4.2 Técnicas de escalonamento 48
4.2.1 Disciplina de serviço FIFO 49
4.2.2 Classificação das disciplinas de serviço 50
4.2.3 Disciplinas de serviço Workconserving 52

4.2.4 Disciplinas de serviço Non-Workconserving 57
4.2.5 Algumas considerações 62
4.2.6 Considerações complementares 65
4.2.7 Conclusões 67
4.3 Algoritmos de consolidação de células de gestão de recursos para conexões ponto-a-multiponto 68
4.3.1 Controlo de congestionamento 68
4.3.2 Propostas 69
4.3.3 Conclusões 74

5 Método para escalonamento e agregação de tráfego nos Dispositivos Periféricos **75**
5.1 Introdução 76
5.2 Pressupostos e objectivos 77
5.3 Arquitectura 78
5.4 Modelo de tráfego 80
5.5 Casos de referência 81
5.6 O Modelo 87
5.7 Caracterização das *curvas de entrada*, de *serviço* e de *saída* do modelo . . 92
5.7.1 *Curva de chegada* 93
5.7.2 *Curva de serviço* 93
5.7.3 Definição da *curva de saída* 94
5.7.4 Relação dos parâmetros prospostos com as curvas 95
5.8 Parametrização do método 96
5.9 Conclusões 100

6 Algoritmos para escalonamento de tráfego **103**
6.1 Primeira solução 104
6.1.1 Solução algorítmica 104
6.1.2 Resultados de simulação 105
6.1.3 Conclusões 113
6.2 Segunda solução 113
6.2.1 Solução intermédia 113

6.2.2 Problemas 114
6.2.3 Solução algorítmica 115
6.2.4 Resultados de simulação 118
6.2.5 Conclusões 124
6.3 Terceira solução 125
6.3.1 Solução intermédia 125
6.3.2 Problemas 127
6.3.3 Solução algorítmica 128
6.3.4 Detalhes de implementação 133
6.3.5 Resultados de simulação 135
6.3.6 Conclusões 139
6.4 Complexidade algorítmica 140
6.5 Resumo e conclusões 142

7 Distribuição de tráfego ABR nos Sistemas Periféricos 145

7.1 Introdução 145
7.2 Critérios de cálculo do peso das fontes na distribuição da largura de banda remanescente 149
7.2.1 1º Critério 150
7.2.2 2º Critério 153
7.2.3 3º Critério 156
7.2.4 4º Critério 158
7.2.5 5º Critério 159
7.2.6 6º Critério 160
7.3 Escolha do critério para o cálculo dos pesos 161
7.3.1 Análise comparativa das várias soluções 161
7.3.2 Exemplo numérico 163
7.3.3 Conclusões 164
7.4 Distribuição da largura de banda remanescente 165
7.4.1 Exemplo de distribuição da largura de banda 167
7.4.2 Solução algorítmica 167
7.5 Resultados de simulação 171

7.5.1 Configuração simples com fontes com diferentes débitos de pico . 171
7.5.2 Configuração complexa com fontes sempre controladas 173
7.5.3 Configuração complexa com fontes com diferentes débitos de pico 175
7.6 Conclusões 176

8 Consolidação de células de gestão de recursos para conexões ponto-a-multiponto **179**
8.1 Introdução 179
8.2 Aspectos a ter em consideração no desenvolvimento de esquemas para conexões ponto-a-multiponto 181
8.3 Definição do modelo 183
8.4 Solução algorítmica 186
8.5 Resultados de simulação 189
8.5.1 Configuração *parking lot* 190
8.5.2 Configuração em *Cadeia* 194
8.5.3 Configuração em *Cadeia* com perturbações 195
8.6 Conclusões 200

9 Conclusões e perspectivas futuras **203**
9.1 Conclusões 203
9.2 Perspectivas futuras 206

Bibliografia **211**

A *Network Calculus* **223**
A.1 Ferramenta matemática 223
A.1.1 Operações 223
A.1.2 Funções de referência 225
A.1.3 Curvas de Chegada 225
A.1.4 Curvas de Serviço 227
A.1.5 Limites 228
A.2 Justificação formal do modelo 229
A.2.1 Definição de novos limites 230
A.2.2 *Curva de Chegada* 231

A.2.3 *Curva de Serviço* 233
A.2.4 Definição da *curva de saída* 236

B Componente *Edge Device* **243**
B.1 Estrutura base de um componente no simulador 243
B.2 Estrutura base do componente ED 245
B.3 Listagem completa do componente *Edge Device* 247

C Avaliação de algoritmos de escalonamento **293**
C.1 Introdução 294
C.2 Disciplina de serviço FIFO 295
C.2.1 Detalhes de implementação 295
C.2.2 Resultados de simulação 295
C.2.3 Conclusões 299
C.3 Disciplina de serviço WFQ 299
C.3.1 Detalhes de implementação 299
C.3.2 Resultados de simulação 300
C.3.3 Conclusões 305

D Algoritmos das fontes e dos destinos **307**
D.1 Comportamento dos componentes 307

Lista de Tabelas

5.1 Parâmetros de configuração para o Cenário 1 85

5.2 Parâmetros de configuração para o Cenário 2 85

5.3 Parâmetros de configuração para o Cenário 3 86

5.4 Parâmetros de configuração para o Cenário 4 86

5.5 Parâmetros de configuração para o Cenário 5 86

5.6 Parâmetros de configuração para o Cenário 6 87

6.1 Resultados numéricos que mostram as deficiências da primeira solução intermédia. 114

7.1 Resumo das propriedades dos vários critérios. 162

7.2 Resultados numéricos obtidos com a aplicação dos vários critérios. 164

7.3 Exemplo de distribuição da largura de banda. 167

7.4 Resultados esperados nos testes de simulação com uma configuração com cinco fontes ABR (com diferentes débitos de pico) e uma VBR. 172

7.5 Resultados esperados nos testes de simulação com a segunda configuração com fontes sempre controladas. 174

7.6 Resultados esperados nos testes de simulação com a segunda configuração com fontes com diferentes débitos de pico. 175

8.1 Parâmetros das fontes e dos comutadores. 189

8.2 Distribuição da largura de banda pelas fontes ABR nos períodos *On* e *Off* da fonte VBR na configuração *Parking lot*. 191

8.3 Distribuição da largura de banda pelas fontes ABR nos períodos *On* e *Off* da fonte VBR na configuração em *Cadeia* com perturbações. 197

A.1 Funções de referência 225

D.1 Parâmetros utilizados nos algoritmos dos componentes 308

Lista de Figuras

2.1 Token bucket com *buffer* na entrada. 10

2.2 Estrutura geral do modelo. 23

2.3 Arquitectura interna do *backbone*. 30

2.4 Indentificação de problemas em VPs do tipo multiponto-a-ponto. 31

3.1 Exemplo de uma curva de chegada e de serviço. 37

3.2 Exemplo de resultados. 43

3.3 Configuração geral utilizada nos testes de simulação. 45

4.1 Classificação dos escalonadores de pacotes. 50

4.2 Chegada de pacotes a um escalonador *Work-conserving*. 62

4.3 Chegada de pacotes a um escalonador *Non-Work-conserving*. 63

4.4 Exemplo de funcionamento da solução SCED e do seu melhoramento para H-FSC. 66

4.5 Configuração ponto-a-multiponto. 69

5.1 Configuração usada para explicar o método. 79

5.2 Arquitectura geral do *Edge Device* (1). 79

5.3 Arquitectura do módulo de agregação e escalonamento dos fluxos IP. . . 80

5.4 Resultados "óptimos" para uma configuração com duas fontes decaladas em um quarto do tempo. 83

5.5 Resultados "óptimos" para uma configuração com duas fontes muito decaladas. 84

5.6 Representação gráfica dos parâmetros do método. 90

5.7 Relação entre os vários parâmetros utilizados no modelo. 91

5.8 Definição dos parâmetros para a parametrização do método. 96

6.1 Resultados obtidos com o primeiro algoritmo no Cenário 1. 106
6.2 Resultados obtidos com o primeiro algoritmo no Cenário 2. 107
6.3 Resultados obtidos com o primeiro algoritmo no Cenário 3. 109
6.4 Resultados obtidos com o primeiro algoritmo no Cenário 4. 110
6.5 Resultados obtidos com o primeiro algoritmo no Cenário 5. 111
6.6 Resultados obtidos com o primeiro algoritmo no Cenário 6. 112
6.7 Intervalos fixos de tamanho k. 116
6.8 Intervalos de tamanho w com $w < k$. 116
6.9 Resultados obtidos com o segundo algoritmo no Cenário 1. 119
6.10 Resultados obtidos com o segundo algoritmo no Cenário 2. 120
6.11 Resultados obtidos com o segundo algoritmo no Cenário 3. 121
6.12 Resultados obtidos com o segundo algoritmo no Cenário 4. 122
6.13 Resultados obtidos com o segundo algoritmo no Cenário 5. 123
6.14 Resultados obtidos com o segundo algoritmo no Cenário 6. 124
6.15 Situações de resolução mais simples. 129
6.16 Situações com resolução contrária. 130
6.17 Exemplo da utilização dos créditos em ambas as situações. 131
6.18 Resultados obtidos com o terceiro algoritmo no Cenário 1. 136
6.19 Resultados obtidos com o terceiro algoritmo no Cenário 2. 137
6.20 Resultados obtidos com o terceiro algoritmo no Cenário 3. 138
6.21 Resultados obtidos com o terceiro algoritmo no Cenário 4. 139
6.22 Resultados obtidos com o terceiro algoritmo no Cenário 5. 140
6.23 Resultados obtidos com o terceiro algoritmo no Cenário 6. 141

7.1 Arquitectura geral do *Edge Device* (2). 146
7.2 Informação utilizada no cálculo do peso de cada fonte. 151
7.3 Configuração com fontes ABR ponto-a-ponto. 172
7.4 Resultados obtidos com a configuração com fontes com diferentes débitos de pico. 173
7.5 Configuração com fontes ABR e VBR ponto-a-ponto. 173

7.6 Resultados do primeiro e segundo comutador na configuração complexa com fontes sempre controladas. 174
7.7 Resultados do primeiro e segundo comutador na configuração complexa com fontes com diferentes débitos de pico. 176

8.1 Configuração *Parking lot.* 191
8.2 Resultados obtidos no primeiro comutador da configuração *Parking Lot.* . . 192
8.3 Resultados obtidos no segundo comutador da configuração *Parking Lot.* . . 193
8.4 Configuração em cadeia. 194
8.5 Resultados obtidos no primeiro e segundo comutador da configuração em *Cadeia.* 195
8.6 Resultados obtidos no terceiro comutador da configuração em *Cadeia.* . . . 196
8.7 Configuração em cadeia com uma fonte VBR. 196
8.8 Resultados obtidos no primeiro comutador da configuração em *Cadeia* com perturbações. 197
8.9 Resultados obtidos no segundo comutador da configuração em *Cadeia* com perturbações. 198
8.10 Resultados obtidos no terceiro comutador da configuração em *Cadeia* com perturbações. 199

A.1 Funções de referência 225
A.2 Definição da curva de serviço. A função B^{out} deve estar acima de $A^{in} \otimes \beta$, que é a menor das envolventes de todas as curvas $A^{in}(t_0) + \beta(t - t_0)$. . . . 228
A.3 Função composta. 238
A.4 Divisão das funções representativas das curvas em duas áreas distintas. . 239
A.5 Definição das variáveis para a definição da *curva de saída.* 240

C.1 Resultados obtidos com a disciplina de serviço FIFO no Cenário 1. . . . 296
C.2 Resultados obtidos com a disciplina de serviço FIFO no Cenário 2. . . . 297
C.3 Resultados obtidos com a disciplina de serviço FIFO no Cenário 3. . . . 298
C.4 Resultados obtidos com a disciplina de serviço FIFO no Cenário 4. . . . 299
C.5 Resultados obtidos com a disciplina de serviço FIFO no Cenário 5. . . . 300
C.6 Resultados obtidos com a disciplina de serviço FIFO no Cenário 6. . . . 301

C.7 Resultados obtidos com a disciplina de serviço WFQ no Cenário 1. . . . 302

C.8 Resultados obtidos com a disciplina de serviço WFQ no Cenário 2. . . . 303

C.9 Resultados obtidos com a disciplina de serviço WFQ no Cenário 3. . . . 304

C.10 Resultados obtidos com a disciplina de serviço WFQ no Cenário 4. . . . 305

C.11 Resultados obtidos com a disciplina de serviço WFQ no Cenário 5. . . . 306

C.12 Resultados obtidos com a disciplina de serviço WFQ no Cenário 6. . . . 306

D.1 Algoritmo da fonte. 309

D.2 Algoritmo do destino. 310

Acrónimos

AAL	ATM Adaptation Layer
AATM	Aplicação ATM
ABR	Available Bit Rate
ACK	Acknowledgments
ACR	Allowed Cell Rate
AF	Assured Forwarding
AIR	Additive Increase Rate Factor
ANSI	American National Standard Institute
ARIS	Aggregate Router-Based IP Switching
ATM	Asynchronous Transfer Mode
ATM TM	ATM Traffic Management
AVF	Average Factor
BE	Best Effort
BECN	Backward Explicit Congestion Notification
BRM	Backward Resource Management
BTE	Broadband Terminal Equipment
CAC	Connectiom Admission Control
CAPC	Congestion Avoidance using Proportional Control
CATM	Comutador ATM
CBR	Constant Bit Rate
CCR	Corrent Cell Rate
CDV	Cell Delay Variation
CDVT	Cell Delay Variation Tolerance
CER	Cell Error Ratio
CL	Controlled Load

CLP	Cell Loss Priority
CLR	Cell Loss Rate
CMR	Cell Misinsertion Rate
CS	Convergence Sublayer
CSR	Cell Switch Router
Delay-EDD	Delay-Earliest-Due-Date
DiffServ	Differentiated Services
DJ	Delay Jitter
DMRCA	Dynamic Max Rate Control Algorithm
DSCP	Differentiated Services Code Point
ECN	Explicit Congestion Notification
ED	Edge Device
EF	Expedited Forward
EFCI	Explicit Forward Congestion Indication
EPD	Early Packet Discard
EPRCA	Enhanced Proportional Rate Control Algorithm
ERICA	Explicit Rate Indication for Congestion Avoidance
ETSI	European Telecommunication Standard Institute
FCFS	First Come First Served
FECN	Forward Explicit Congestion Notification
FFQ	Fluid Fair Queueing
FIFO	First In First Out
FRM	Forward Resource Management
FRTT	Fixed Round Trip Time
FTP	File Transfer Protocol
GFC	Generic Flow Control
GPS	Generalized Processor Sharing
GS	Guaranteed Service
H-FSC	Hierarchical Fair Service Curve
HRR	Hierarchical Round Robin

ICMP	Internet Control Message Protocol
ICR	Initial Cell Rate
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internetworking Protocol
IPmTB	fluxo IP modelado por *Token Bucket*
ISP	Internet Service Provider
ITU-T	International Telecommunications Union-T
LAN	Local Area Network
LANE	LAN Emulation
LF	Linha Fisica
LIJ	Leaf Initiated Join
LQF	Longest Queue First
LTRB	Least Time to Reach Bound
maxCTD	Maximum Cell Transfer Delay
MBS	Maximum Burst Size
MCR	Minimum Cell Rate
MER	Minimum Explicit Rate
MH	Multiple Hop
MIT	Massachusetts Institute of Technology
MPLS	Multi-Protocol Label Switching
MPOA	Multi-Protocol Over ATM
NHRP	Next Hop Resolution Protocol
NIST	National Institute of Standards and Technology
Nrm	Number of RM cells
nrt-VBR	non-real-time-Variable Bit Rate
N-WCing	Non-Work-Conserving
OSI	Open Systems Interconnection
PCR	Peak Cell Rate
PDA	Personal Data Assistant
PFQ	Packet Fair Queueing

PHB	Per-Hop Behavior
PNNI	Private Network to Node Interface
pp-CDV	Peak-to-Peak Cell Delay Variation
PPD	Partial Packet Discard
PRCA	Proportional Rate Control Algorithm
Q_H	High Threshold
Q_L	Low Threshold
QoS	Quality of Service
RCSP	Rate Control Static Priority Queueing
RDF	Reduction Factor
RDIS	Rede Digital com Integração de Serviços
RDIS-BL	RDIS de Banda Larga
RED	Random Early Detection
RJ	Rate Jitter
RM	Resource Management
RMIP	Resource Management IP
rt-VBR	real-time-Variable Bit Rate
RTD	Round Trip Delay
RTT	Round Trip Time
RSVP	ReSerVation Protocol
SCED	Service Curve Earliest Deadline
SCFQ	Self-Clocked Fair Queueing
SCR	Sustained Cell Rate
SDH	Synchronous Digital Hierarchy
SECBR	Severely Errored Cell Block Ratio
SEAM	Scalable and Efficient ATM Multicast
SH	Single Hop
SMART	Shared Many-to-Many ATM Reservation
TB	Token Bucket
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TM	Traffic Management

UBR	Unspecified Bit Rate
UCSC	University of California Santa Cruz
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunication Systems
UNI	User-to-Network Interface
UPC	Usage Parameter Control
VBR	Variable Bit Rate
VC	Virtual Connection
VCI	Virtual Connection Identification
VoD	Video on-demand
VoIP	Voice over IP
VP	Virtual Path
VPI	Virtual Path Identification
WAN	Wide Area Network
WAP	Wireless Application Protocol
WCing	Work-Conserving
WFQ	Weigthed Fair Queueing
WF^2Q	Worst case Fair Weigthed Fair Queueing
WF^2Q+	Worst case Fair Weigthed Fair Queueing plus

Capítulo 1

Introdução

"As we enjoy great advantages from inventions of others, we should be glad of an opportunity to serve others by any invention of ours; and this we should do freely and generously."

Benjamin Franklin

1.1 Enquadramento

O crescimento explosivo da Internet a par da implantação da tecnologia ATM (*Asynchronous Transfer Mode*) constituem dois exemplos paradigmáticos das profundas transformações por que estão a passar as redes de telecomunicações, assistindo-se actualmente à convergência das redes IP com as redes móveis de última geração.

Se por um lado a Internet assentou o seu sucesso num modelo de serviço do tipo *best effort* (datagrama) vocacionado para a comunicação de dados ponto-a-ponto entre computadores, verifica-se que tráfego com requisitos de tempo real (nomeadamente em aplicações multimédia interactivas) começa a ser actualmente transportado na Internet, embora com limitações óbvias, o que exige extensões ao modelo de serviço tradicional de forma a incluir suporte de qualidade de serviço (QoS).

O crescimento das redes de comunicação tem sido exponencial e pode ser considerado em três vertentes: (i) no número de utilizadores/máquinas; (ii) no número de

subredes interligadas e (iii) no volume de dados transportado (educação, negócios, entretenimento, etc.). Estas redes utilizam um conjunto heterogéneo de tecnologias/meios de transmissão, desde as tradicionais redes de cobre com *modems* a baixa velocidade até às redes em fibra óptica, passando pelas redes de satélites. O desenvolvimento de novos dispositivos terminais e aplicações (telemóveis com acesso à Internet via *Wireless Application Protocol* - WAP, *Universal Mobile Telecommunications Systems* - UMTS, até aos *Personal Data Assistants* - PDAs com ligação à Internet) implica a necessidade de redes com capacidades para assegurar diferentes requisitos de QoS.

Por outro lado, o facto das redes ATM terem sido concebidas para transportar uma grande diversidade de tipos de tráfego (aúdio, vídeo e dados) faz com que os fornecedores de serviços IP (*Internet Service Providers*, ISPs) vejam as redes ATM como uma componente de soluções capazes de suportar aplicações que necessitam de garantias de qualidade de serviço (largura de banda e controlo dos atrasos). Do ponto de vista tecnológico, as redes ATM baseiam-se na comutação rápida de pacotes de comprimento fixo e pequeno (células), segundo um modo orientado às conexões (circuitos virtuais); do ponto de vista arquitectónico caracterizam-se por um conjunto de protocolos e mecanismos de controlo orientados para o suporte de diferentes níveis de QoS. Estes diferentes níveis são assegurados com base na definição de categorias de serviço que têm como objectivo relacionar parâmetros de tráfego e de QoS com o comportamento da rede de forma a conjugar a utilização eficiente de recursos com o cumprimento dos contratos negociados pelos utilizadores. Esta tecnologia aparece como uma solução "apetecível" devido às suas capacidades em resolver os problemas referidos anteriormente, sendo em particular adequada para infra-estruturas de transporte de grande dimensão (*backbones*).

Independentemente da tecnologia utilizada (IP, ATM ou outra), quando as aplicações são "críticas" devem ter uma prioridade superior às aplicações de dados elásticas, como por exemplo sessões FTP ou serviços de correio electrónico, uma vez que as primeiras necessitam de garantias de desempenho relacionadas com o atraso e com a sua variação. Assim é necessário tomar decisões na forma como o tráfego é gerido na rede, isto é, dar tratamento preferencial (reserva de recursos e escalonamento) a um tipo de tráfego em detrimento de outro com requisitos menos exigentes (*best effort*, por exemplo). Como consequência desta separação e das diferentes prioridades a observar torna-se necessário definir categorias de serviço tendo em atenção classes de aplicações

actuais e futuras. O surgimento de novas aplicações cria necessidades cada vez mais complexas relativamente à gestão do tráfego. Em comunicações realizadas numa filosofia ponto-a-ponto, os problemas são já conhecidos e as soluções existentes são suficientemente robustas para os resolver. Contudo, com o aparecimento de aplicações de aúdio ou vídeo-conferência, ensino à distância, aplicações distribuídas, etc., (aplicações que necessitam de comunicações multiponto) os problemas referidos apresentam outro tipo de complexidade.

Como referido anteriormente, com o crescimento do nível de exigência das aplicações a melhor forma de tirar partido de várias tecnologias é estruturá-las com o objectivo de garantir às aplicações o melhor serviço possível. Assim, a tradicional separação das redes de telecomunicações com base nos serviços que disponibilizam tem vindo a ser substituída pelo conceito de integração de serviços numa única rede - e deste ponto de vista as tecnologias de comutação (rápida) de pacotes, baseadas em ATM ou IP, pela sua flexibilidade e escalabilidade, revelam-se como possíveis soluções.

Com base nesta ideia e com o aumento substancial do número de aplicações com requisitos cada vez mais exigentes que utilizam redes IP, uma das soluções é transportar este tipo de tráfego em *backbones* de alta velocidade, estruturados com base numa rede de núcleo (utilizando uma tecnologia de comutação de etiquetas, por exemplo o ATM, o *Multi-Protocol Label Switching* (MPLS), etc.) ligando dispositivos periféricos (*Edge Devices*, ED) que constituem a *interface* a diversas redes de acesso (baseadas em diferentes tecnologias e modelos de comunicação). Com esta organização é possível transferir para a periferia da rede (para os ED) funções críticas como a classificação e a agregação do tráfego em classes com requisitos bem definidos, deixando para a rede de núcleo a comutação rápida com um mínimo de processamento possível, o que permite aumentar a velocidade de transferência (redução da latência) e fornecer soluções escaláveis.

Os modelos de QoS existentes quer em ATM quer em IP têm percursos diferentes devido à forma como foram desenvolvidos. Enquanto que o modelo de QoS em ATM foi estabelecido desde o início como um objectivo central, o modelo de QoS em IP foi desenvolvido à medida que as aplicações o exigiam. Numa primeira fase foi desenvolvido o *IntServ* [103, 78, 6] com um protocolo de reserva de recursos associado, mas mostrou-se "pesado". Mais tarde, e com o objectivo de resolver os problemas do modelo de QoS anterior, apareceu o *DiffServ* [32], que é neste momento o modelo de QoS em IP mais

discutido.

Uma vez que o modelo de serviço de QoS em IP, *DiffServ*, se baseia na ideia de agregação de tráfego e funcionalidades reduzidas nos nós, e com o objectivo de transportar na rede de núcleo fluxos agregados, a utilização do ATM ou do MPLS apresentam-se como os possíveis candidatos ao suporte de serviços diferenciados. É no entanto necessário destacar o papel importante que os EDs vão desempenhar relativamente a esta função, isto é, do lado da rede de acesso recebem fluxos individuais ou com um nível de agregação reduzido, enquanto que na rede de núcleo apenas existem fluxos agregados tratados no contexto dum número reduzido de classes. Como o tráfego agregado deixa de poder ser controlado individualmente na rede de núcleo é importante escalonar os fluxos de forma criteriosa antes de os agregar. Assim, os EDs devem controlar o tráfego através de mecanismos de policiamento e de modelação (*shaping*), para que seja possível garantir a QoS negociada inicialmente e, para além disso, suportar disciplinas de serviço (algoritmos de escalonamento) para uma adequada distribuição da largura de banda disponível pelas classes previamente definidas (de acordo com critérios rigorosos).

Tanto em IP como em ATM existem classes de serviço que têm o seu tráfego caracterizado por três parâmetros: o débito máximo instantâneo, o débito médio garantido e o tamanho máximo dos *bursts*. Dada a sua importância, este é o tipo de tráfego utilizado na tese, sendo objecto de análise os problemas da distribuição quanto aos atrasos e quanto ao controlo da ocupação dos *buffers*. Estes problemas serão avaliados em disciplinas de serviço conhecidas e serão propostas novas disciplinas. Na proposta de novas disciplinas de serviço é necessário ter em consideração que o objectivo principal é o escalonamento e este implica contenção quando algum recurso é partilhado. O escalonamento é realizado com base em critérios adequados na tomada de decisão sobre qual a célula ou pacote que vai ser servido a seguir, de forma a garantir a QoS negociada. A garantia da QoS pode ser dada com base na reserva da largura de banda ou controlo dos atrasos. Estas garantias e a forma como são obtidas indicam e determinam o melhor ou pior desempenho da rede na gestão dos seus recursos.

A par dos problemas discutidos anteriormente (agregação e escalonamento na periferia) é necessário discutir qual a tecnologia apropriada para o transporte de fluxos agregados no interior da rede. Para garantir os requisitos dos fluxos agregados quanto à QoS é defendido que estes fluxos podem ser transportados na rede de núcleo usando

uma categoria de serviço com propriedades idênticas à categoria de serviço *Available Bit Rate* (ABR) [44, 42] (isto é, no caso da rede de núcleo utilizar ATM, ou idêntica a ABR se for usada outra tecnologia de comutação). No caso de ser usada a categoria de serviço ABR, é necessário tratar outro tipo de problemas, nomeadamente o controlo de tráfego e de congestionamento para as conexões ponto-a-ponto e ponto-a-multiponto. Relativamente às conexões ponto-a-ponto o trabalho realizado na tese de mestrado do autor [81] apresentou uma solução para este problema; no entanto, para as conexões ponto-a-multiponto não existe ainda um *standard*, sendo as recomendações do ATM Forum escassas.

1.2 Objectivos e principais contribuições da tese

1.2.1 Objectivos

O trabalho iniciou-se com uma pesquisa bibliográfica aprofundada e actualizada e a análise crítica de algoritmos de gestão de tráfego ponto-a-ponto e ponto-a-multiponto em ATM, com especial ênfase nos problemas da distribuição dos recursos (largura de banda), incluindo o acompanhamento das actividades de normalização do IETF e do ATM Forum; os objectivos essenciais desta fase foram a sistematização dos problemas e a compreensão e clarificação das relações entre eles.

Numa segunda fase foi identificado o conjunto de problemas específicos a abordar e a respectiva metodologia; foi dada ênfase especial ao problema da reserva de recursos com base na QoS. Foram explorados e comparados diversos modelos descritos na literatura.

Como objectivo final propõe-se a definição de uma arquitectura para o equipamento periférico e um modelo de escalonamento com estratégias de controlo associadas, que permitam integrar (a nível de serviços) os ambientes ATM e Internet; foram igualmente definidos critérios e cenários de validação do modelo e realizada a respectiva avaliação de desempenho recorrendo a técnicas de simulação. Neste modelo estão incluídas disciplinas de serviço capazes de gerir recursos (largura de banda), de forma a que os atrasos sejam distribuídos pelos fluxos de tráfego da classe em análise. Este tipo de fluxos são transportados na rede de núcleo através da utilização da categoria de serviço ABR. É ainda estudada a forma como os recursos são distribuídos dentro dos

EDs quando as conexões ATM são do tipo ponto-a-ponto e ponto-a-multiponto, bem como são tratados e resolvidos os problemas típicos de conexões multiponto no interior da rede ATM.

1.2.2 Contribuições da tese

De acordo com os objectivos definidos anteriormente para este trabalho são apresentadas quatro contribuições que permitiram atingir os referidos objectivos.

Em primeiro lugar, o facto de existir um número crescente de aplicações IP com requisitos cada vez mais exigentes, e do tipo de solução escolhida para a estruturação da rede, foi proposta uma arquitectura para o ED de forma a agregar os fluxos da classe de tráfego em análise sobre ATM usando VCs.

Dos modelos de QoS em IP, o *DiffServ* utiliza a agregação de fluxos, e como a ideia é transportar os fluxos agregados na rede de núcleo (permitindo uma melhor gestão dos recursos e aumentando a escalabilidade), é proposta uma nova disciplina de serviço para a distribuição dos recursos pelo fluxos IP dentro dos EDs, de forma a garantir uma distribuição da largura de banda, conseguindo equidade na distribuição dos atrasos e da ocupação dos *buffers*.

Como a escolha para o transporte dos fluxos agregados na rede de núcleo incidiu em VCs do tipo ABR, surgiu a questão da distribuição da largura de banda por VCs deste tipo dentro dos EDs. Para a resolução desta questão foi proposto um esquema de distribuição equitativa da largura de banda por VCs ABR, no caso de negociarem a atribuição de uma largura de banda mínima garantida.

Para o transporte dos fluxos agregados era possível a utilização de VCs ABR do tipo ponto-a-ponto ou multiponto, mas como para o primeiro caso já existia uma solução proposta pelo autor [81] para a gestão de recursos, decidiu-se estender os algoritmos de gestão de recursos deste tipo de VCs para VCs multiponto, através da proposta de um algoritmo de consolidação de células de gestão de recursos para conexões ponto-a-multiponto.

1.3 Organização da tese

No Capítulo 2 é realizada uma abordagem aos modelos de QoS das duas tecnologias referidas (IP e ATM), bem como as tendências actuais de integração. Houve a preocupação de fornecer neste capítulo o maior número de referências possível, para que o leitor possa, se necessário, recorrer aos artigos e relatórios técnicos mais pormenorizados sobre cada um dos tópicos cobertos. À medida que os assuntos são apresentados, caracterizam-se também os problemas que serão objecto de estudo neste trabalho.

Os aspectos relacionados com as ferramentas de análise e simulação são abordados no Capítulo 3. Neste capítulo é introduzida uma ferramenta matemática amplamente usada na literatura que serviu para validar o modelo de escalonamento proposto. São ainda discutidas questões relacionadas com as técnicas de simulação. É apresentado o simulador utilizado na realização dos testes, não esquecendo todas as modificações necessárias para que este pudesse ser executado em máquinas diferentes daquela para a qual foi pensado, nomeadamente a construção de raiz de um componente para simular o comportamento dos EDs. São ainda descritas duas pequenas aplicações concebidas para ajudar na preparação dos dados para o posterior tratamento estatístico.

O principal objectivo do Capítulo 4 é apresentar os trabalhos mais significativos das várias áreas tratadas (algoritmos de escalonamento e algoritmos de consolidação de células de gestão de recursos e de distribuição da largura de banda em conexões ponto-a-muliponto) de forma a ilustrar os problemas.

No Capítulo 5 é desenvolvido um método de escalonamento para tráfego com requisitos de QoS, acompanhado de uma justificação formal com base na ferramenta matemática apresentada no Capítulo 3.

Os algoritmos de suporte ao método de escalonamento são apresentados no Capítulo 6, sendo a respectiva avaliação e justificação realizada através de testes de simulação em configurações previamente definidas.

No Capítulo 7 é proposto um algoritmo para a distribuição da largura de banda, dentro do ED, com base num critério proposto para o cálculo de pesos a atribuir aos VCs ABR.

No Capítulo 8 é apresentada uma solução para os problemas típicos das conexões ponto-a-multiponto, com especial ênfase na forma como o congestionamento é tratado

quando ocorre.

No Capítulo 9 são apresentadas as conclusões do trabalho, dando especial destaque às principais contribuições; são também sugeridas algumas pistas para trabalhos futuros.

No Anexo A é apresentada detalhadamente a ferramenta matemática *Network Calculus*, bem como a justificação formal do modelo desenvolvido.

O Anexo B apresenta a estrutura do componente *Edge Device*, acompanhada da respectiva listagem do código.

No Anexo C são apresentados e discutidos os resultados de simulação obtidos em configurações de referência. Os testes foram realizados com a disciplina mais simples, *First Come First Served* (FCFS), e uma disciplina referida na literatura como uma das mais eficientes, nomeadamente *Weigthed Fair Queueing* (WFQ), com o objectivo de perceber quais as suas debilidades.

O Anexo D apresenta os algoritmos da fonte e do destino para a categoria de serviço ABR, uma tabela com a definição dos parâmetros utilizados, bem como uma classificação dos algoritmos implementados nos comutadores (para o controlo dos fluxos e do congestionamento) utilizados pelo ATM Forum.

Capítulo 2

Enquadramento geral e caracterização de problemas

A utilização de duas tecnologias (neste caso, uma tecnologia de comutação de etiquetas e o IP) com diferentes características, levanta um conjunto de problemas com especificidades e realidades próprias que conduzem a um trabalho adicional, quando se pretende que ambas coexistam. Com o propósito de fornecer Qualidade de Serviço (QoS) através da coexistência de ambas as tecnologias, e de tirar partido das suas vantagens, é necessário apresentar características e identificar pontos em comum e de fácil convergência, para poder estabelecer essa relação, para além de mostrar os seus problemas.

O tema relacionado com a QoS é objecto da primeira secção, onde são abordados os aspectos a ter em consideração, enquanto que na segunda secção são apresentadas resumidamente as arquitecturas e soluções de coexistência entre a tecnologia IP sobre ATM. Na terceira secção é apresentada a tendência actual de integração das duas tecnologias e são caracterizados os seus problemas, enquanto que na última secção são apresentados e discutidos os problemas relativos às conexões multiponto em redes ATM.

2.1 Modelos de Qualidade de Serviço

A QoS é uma área crescente de investigação e desenvolvimento em redes de computadores, devido à nova geração de aplicação que utiliza a Internet, uma vez que algumas das tecnologias usadas não foram pensadas para suportar tráfego com prioridades ou garantir diferentes níveis de desempenho, tornando difícil implementar soluções para

garantir QoS.

A QoS pode ser definida em termos da rede na sua globalidade ou em termos do desempenho de um servidor ou de um *router*. Apesar de não existir uma definição formal de QoS existem várias definições que acabam por convergir para uma, que se pode resumir na disponibilidade da rede em garantir resultados previsíveis; isto é, das definições encontradas na literatura (X.902 do ITU ou o léxico do ATM) todas apontam para um modelo de serviço que dê garantias quanto à largura de banda (*throughput*), à latência (*delay*) e à taxa de erros.

2.1.1 Caracterização e modelação de tráfego

A caracterização e a modelação do tráfego tem um papel muito importante para a obtenção de bons índices de desempenho de uma rede, uma vez que o que se conhece *à priori* é mais fácil de tratar. Assim, é possível destacar alguns dos modelos mais populares na caracterização do tráfego (encontrados em quase todos os artigos relacionados com disciplinas de serviço): Poisson [57], On-Off [57], *Markoviano* (modelo utilizado para caracterizar fontes de vídeo [92]), $(X_{min}, X_{ave}, I, S_{max})$ [38], (σ, ρ) [25] e (r, T) [66]. Para se ter uma ideia mais precisa sobre a caracterização do tráfego é possível, por exemplo analisar o caso da Internet; quando existe a necessidade de fornecer QoS um dos modelos usados é o *Integrated Services* [103, 104], que permite caracterizar o tráfego através dos parâmetros (p, M, r, b) com p a representar o débito de pico, M o tamanho máximo do pacote, r o débito médio e b o tamanho máximo do *burst*.

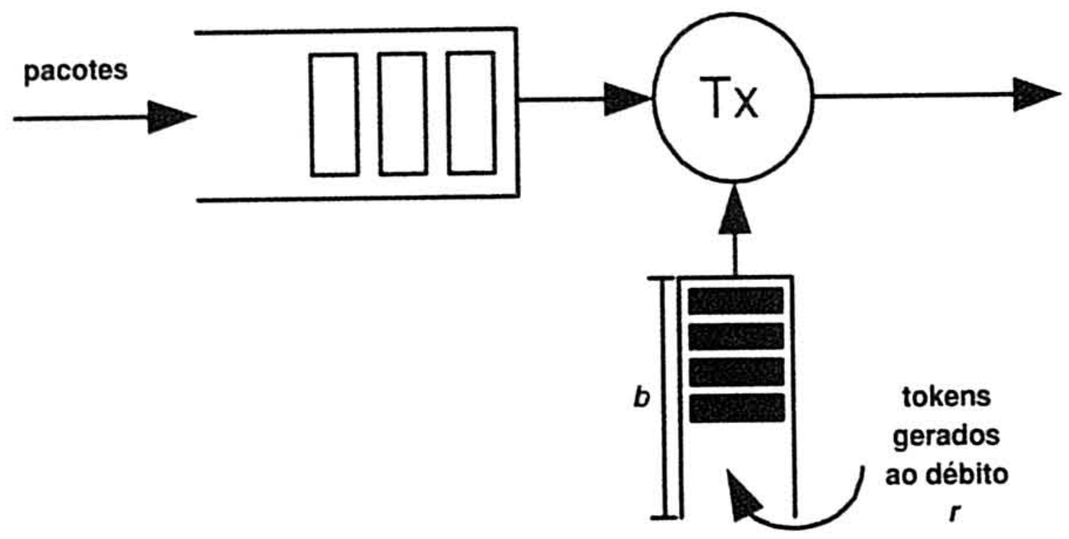

Figura 2.1: Token bucket com *buffer* na entrada.

Estando o tráfego caracterizado é necessário modelar e/ou policiar o tráfego na entrada da rede, através de um mecanismo apropriado, onde se pode destacar o *Token Bucket* [53, 113, 115], que fornece limites ao padrão de tráfego [10, 25, 27] que pode ser

transmitido. Um TB (ver Figura 2.1) é basicamente um mecanismo de gestão de créditos que controla a entrada do tráfego na rede. O seu modo de funcionamento caracteriza-se pela entrada de *tokens* no *bucket* de tamanho b com um débito r (todos os *tokens* gerados que ultrapassem a capacidade do *bucket* são eliminados), e sempre que chega um pacote[1] e existe um *token* disponível, o pacote entra na rede e o número de *tokens* é decrementado; por outro lado, se chega um pacote e não existem *tokens* disponíveis duas situações podem ocorrer, de acordo com a implementação utilizada: se o TB tem um *buffer* a montante os pacotes esperam pela disponibilidade de *tokens* (os pacotes são atrasados) ou são marcados e transmitidos com uma prioridade baixa e em momentos de congestionamento são os primeiros a serem eliminados, ou ainda se não existe nenhum *buffer* para armazenar os pacotes estes são simplesmente eliminados.

Após os pacotes serem admitidos na rede eles podem ainda ser eliminados num qualquer equipamento ao longo do percurso, quando encontram *buffers* cheios, ou então por alguma redução momentânea da largura de banda devido a um elevado congestionamento. Esta situação levanta a questão da importância da escolha óptima dos parâmetros do TB como discutido em [72, 92, 10, 25, 27, 113, 115, 47].

Como conclusão, um regulador em geral, e um TB em particular, pode ser visto como um veículo para a transferência do congestionamento e da perda de tráfego do interior para a periferia da rede, de forma a que as perdas e o congestionamento possam ser antecipados e assim permitir tomar as acções necessárias.

2.1.2 Aspectos a considerar em modelos de QoS

Antes de apresentar os modelos de QoS em ATM e em IP é necessário introduzir algumas noções importantes e assuntos relacionados com o fornecimento de QoS.

A QoS de uma rede pode ser definida de várias formas e incluir o seguinte conjunto de requisitos de serviço: o *desempenho*, a *disponibilidade*, a *fiabilidade* e a *segurança*. Todos estes requisitos de serviços são aspectos importantes no fornecimento de QoS por parte da rede. Contudo, é necessário estabelecer métricas de desempenho usadas na definição de QoS da rede, tais como a *largura de banda*, o *atraso*, a *variação do*

[1] Os pacotes podem ser de comprimento fixo ou variável. No caso dos pacotes de comprimento variável pode ser consumido mais do que um *token* e, por isso serão utilizados, por questões de simplicidade de exposição, pacotes de comprimento fixo, uma vez que por cada pacote deste tipo é consumido apenas um *token*.

atraso (*jitter*) e a *percentagem de pacotes perdidos*. Usando estas métricas, as garantias de desempenho da rede podem ser especificadas de forma *absoluta* (por exemplo, a rede garante 5 $Mbit/s$ a uma conexão durante o seu tempo de vida), ou *probabilística* (por exemplo, a rede garante um atraso inferior a 100 ms para 90% dos pacotes). A característica de garantia de QoS numa rede é que diferencia estas redes das redes que fornecem serviços do tipo *best-effort*.

Para que as redes forneçam garantias de QoS existem dois factores que são determinantes, o primeiro para aplicações com requisitos rígidos de QoS, em que podem destacar-se as aplicações de VoIP e vídeo a pedido (*Video on-demand*, VoD), sendo que no primeiro caso (mais rígido do que o segundo) depende das garantias relativamente a um reduzido atraso e uma reduzida percentagem de perdas, enquanto que para o segundo caso necessita de garantias relativamente à largura de banda e à percentagem de pacotes perdidos. O segundo factor tem a ver com a necessidade de ter serviços diferenciados, isto é, a possibilidade de ter várias combinações de garantias por parte dos fornecedores de serviços.

O fornecimento de garantias de QoS requer o desenvolvimento de mecanismos de controlo apropriados. Para compreender estes mecanismos e a sua complexidade é necessário considerar vários aspectos do controlo. Um dos aspectos mais importantes no controlo de QoS é a *escala de tempo* a que o mecanismo opera, sendo possível dividir esta escala em quatro níveis:

- **Nível do pacote**: A unidade mais pequena sobre a qual a rede pode exercer controlo. Mecanismos que operam a este nível incluem controladores de tráfego (classificadores de tráfego, marcadores de tráfego, policiadores e modeladores), escalonadores de pacotes e mecanismos de gestão de filas de espera.

- **Nível do Round-trip-time**: Mecanismos de controlo de QoS baseados em *feedback*, tais como controlo de fluxo e de congestionamento.

- **Nível da sessão**: Mecanismos de controlo de QoS, tais como o controlo de admissão e de *routing*.

- A um **nível superior** (horas, dias, etc.): É possível encontrar a engenharia de tráfego, a política de tarifação, o planeamento, etc.

Outro dos aspectos importantes do controlo de QoS é a **granularidade** da informação de controlo (o estado do controlo) usado por um mecanismo de QoS, quando toma decisões e exerce controlo. A granularidade pode ser dividida em granularidade "fina" quando é usado controlo sobre a informação de estado por fluxo para garantir QoS para utilizadores individuais, ou granularidade "grossa" quando a informação é específica e mantida por agregado de fluxos (esta granularidade varia dependendo do nível de agregação dos fluxos, por *host*, por prefixo, por classe de serviço, etc.). Relacionado com este aspecto do controlo é ainda necessário, por um lado, decidir onde é armazenado o estado do controlo (nos *routers* ou apenas no cabeçalho dos pacotes) e, por outro lado, onde está localizado o controlo, isto é, onde o mecanismo de controlo opera (nos sistemas finais, na periferia da rede, nas ligações entre utilizadores e a rede, entre domínios de rede, etc.).

Com base na combinação destes factores é possível conceber uma grande variedade de arquitecturas com vários graus de complexidade. Por exemplo, uma arquitectura que fornece QoS, que exerce controlo por fluxo e armazena estado de QoS em cada um dos *routers*, necessita de um protocolo de sinalização que forneça o estado de QoS a todos os *routers* e por fluxo. Como se pode observar esta arquitectura exige uma solução relativamente sofisticada em cada *router*, complica a sua operação e limita a sua escalibilidade. Por outro lado, para conceber uma arquitectura escalável é necessário tomar decisões conscientes relativamente às opções quanto à *escala de tempo* utilizada e à *granularidade* pretendida, bem como à *localização* do armazenamento da informação de controlo e dos respectivos mecanismos e avaliar cuidadosamente os seus compromissos e implicações.

2.1.3 Modelo de QoS em ATM

As redes ATM têm necessidade de mecanismos de gestão de tráfego não só para garantirem a entrega das células com um determinado nível de qualidade, mas também com o objectivo de maximizar a utilização dos recursos da rede. Estes mecanismos possibilitam que o utilizador especifique um conjunto de parâmetros que estarão directamente relacionados com a qualidade de serviço desejada, sendo estes conhecidos por parâmetros de QoS.

A par dos parâmetros de QoS é necessário ter parâmetros que caracterizem o

tráfego que vai ser transportado na rede de acordo com a categoria de serviço escolhida. Após o estabelecimento da conexão na respectiva categoria, o fluxo de dados à entrada da rede vai ser controlado através de mecanismos de policiamento e de modelação (*shaping*), de forma a garantir que o contrato de tráfego acordado seja cumprido de parte a parte.

Parâmetros de QoS

Os valores dos parâmetros de QoS são escolhidos de acordo com o tipo de serviço que é suportado. Estes parâmetros caracterizam o desempenho de uma conexão na camada ATM e quantificam o desempenho da rede numa base extremo-a-extremo[2]. Os parâmetros definidos são em número de sete, sendo quatro negociáveis entre a rede e os utilizadores e os restantes não negociáveis. Os parâmetros negociáveis são os seguintes:

- *Cell Loss Ratio* (CLR): percentagem de células perdidas na rede, por exemplo, devido ao congestionamento, uma vez que a rede elimina preferencialmente as células que tenham atribuído o valor 1 ao *bit Cell Loss Priority* (CLP) no cabeçalho das células, enquanto que se o *bit* CLP é zero, as células só serão eliminadas pela rede se existirem problemas persistentes que os algoritmos de controlo de tráfego não consigam resolver.

- *Maximum Cell Transfere Delay* (maxCTD): representa o atraso máximo de uma célula entre o ponto de entrada e o de saída, designado por atraso de transferência. Este atraso inclui o atraso de propagação, atrasos nas filas de espera e o tempo de serviço em cada fila.

- *Cell Delay Variation* (CDV): descreve a variabilidade no padrão de chegada das células, designado na literatura por *"delay jitter"*. O *jitter* deve ser limitado de acordo com os objectivos do serviço e os *buffers* são dimensionados com base nos valores pretendidos.

- *Peak-to-Peak Cell Delay Variation* (pp-CDV): o termo *peak-to-peak* refere-se à diferença entre o *maxCTD* e o *CTDmin*, correspondendo às componentes fixas do atraso na rede.

Os parâmetros não negociados são os seguintes:

[2]Os parâmetros de tráfego e da QoS estão descritos detalhadamente em [41].

- *Cell Error Ratio* (CER)
- *Severely Errored Cell Block Ratio* (SECBR)
- *Cell Misinsertion Rate* (CMR)

Parâmetros de tráfego

Estes parâmetros fornecem as características do tráfego de uma conexão, podendo ser quantitativos ou qualitativos e têm a seguinte definição:

- *Peak Cell Rate* (PCR): é o débito máximo a que os sistemas finais desejam transmitir (definido como o inverso do intervalo mínimo entre células geradas na *interface* entre as camadas ATM e Física).
- *Sustained Cell Rate* (SCR): limite superior do débito médio durante a ligação, mas serve como referência em intervalos de tempo mais curtos (como forma de eliminar a duração de *bursts* transmitidos com débitos superiores).
- *Minimum Cell Rate* (MCR): débito mínimo assegurado ou garantido.
- *Cell Delay Variation Tolerance* (CDVT): Está associado ao PCR, visto que o intervalo entre células observado na ligação física pode ser inferior ao débito de transmissão (1/PCR).
- *Maximum Burst Size* (MBS): Número máximo de células (*burst*) que podem ser transmitidas ao débito máximo.

Categorias de serviço

As categorias de serviço definidas nas redes ATM abrangem um leque muito diversificado de aplicações. Algumas destas aplicações necessitam de garantias por parte da rede para satisfazer um conjunto de requisitos associados. As garantias podem ser obtidas de acordo com as seguintes categorias:

- **Constant Bit Rate**

  As capacidades de transferência da categoria *Constant Bit Rate* (CBR) são usadas em conexões que requerem uma largura de banda constante durante o tempo de

vida da conexão. Este requisito é indicado pelo valor do parâmetro PCR e pelos parâmetros CTD e CDV relativos ao atraso e o CLR relativo às perdas [41]. A partir do momento em que a conexão é estabelecida, os parâmetros QoS negociados são assegurados para todas as células que são transmitidas. A fonte pode não transmitir continuamente com o débito igual a PCR, mas a capacidade correspondente fica reservada.

- **Variable Bit Rate**

  A categoria de serviço *Variable Bit Rate* (VBR) permite ao utilizador transmitir com um débito variável, podendo ser explorada a multiplexagem estatística. Dependendo da aplicação ser ou não sensível ao atraso das células, esta categoria é subdividida em duas:

  - VBR com requisitos de tempo real (rt-VBR): é adequado para aplicações que necessitam de garantias relativamente ao atraso máximo e às variações do atraso [41]. As conexões deste tipo são caracterizadas pelos parâmetros PCR, SCR, MBS, CLR, *max*CTD e ppCDV.

  - VBR sem requisitos de tempo real (nrt-VBR): nesta categoria as fontes caracterizam-se pelos parâmetros PCR, SCR, MBS e CLR. Uma das diferenças entre esta categoria e a anterior é a não existência de limites na variação dos atrasos; no entanto, são necessárias algumas garantias [41]. Esta categoria de serviço permite realizar multiplexagem estatística das várias conexões e não tem associado nenhum limite de atraso. É "*open loop*" com um contrato de tráfego que pode ser realizado na fonte através da utilização de um modelador de tráfego, e "forçado" pela rede e pelo destino através do *Usage Parameter Control* (UPC). É assumido que os recursos são reservados estatisticamente nos nós da rede para satisfazer os contratos de tráfego e garantir a QoS.

- **Unspecified Bit Rate**

  A classe *Unspecified Bit Rate* (UBR) está definida para aplicações de dados que requerem largura de banda variável, mas que não são sensíveis à perda de células ou a atrasos variáveis, pois as aplicações têm os seus próprios mecanismos de

recuperação e de retransmissão, ou podem suportar-se num protocolo dotado com esses mecanismos [41].

- **Available Bit Rate**

  A categoria de serviço *Available Bit Rate* (ABR) tem características bastante específicas relativamente ao controlo de fluxo. É uma categoria de serviço em que as características de transferência dependem da informação de controlo que a rede fornece, uma vez que a largura de banda atribuída a cada conexão varia de forma dinâmica. Os mecanismos de controlo de fluxo aplicáveis a esta categoria envolvem diferentes tipos de *feedback* para controlar o débito das fontes. Este *feedback* é conseguido à custa de uma célula especial designada por célula de gestão de recursos (*Resource Management*, RM) [41, 81], que se designa por *Forward* RM (FRM) quando segue em direcção ao destino e *Backward* RM (BRM) quando é devolvida pelo destino e viaja em direcção à fonte. A política de distribuição da largura de banda pelos VCs deste tipo de serviço tem usualmente como base o critério *max-min fairness* [18]. No estabelecimento de uma conexão ABR, a fonte deve indicar à rede os valores extremos entre os quais quer transmitir [58]. O débito definido em PCR é o débito máximo desejado pela fonte, sendo também possível definir o valor do débito mínimo (MCR) [98], que deve ser garantido pela rede se o débito instantâneo da fonte for superior ou igual ao valor do MCR.

Mecanismos de controlo de tráfego

Os mecanismos de controlo de tráfego fornecidos pelo ATM (*Connection Admission Control*, *Traffic Shaping*, etc.) servem, por um lado para controlar o acesso à rede e, por outro lado, para controlar o congestionamento de forma a minimizar a sua intensidade e duração.

O mecanismo de controlo de admissão (*Connection Admission Control*, CAC) fornecido pelo ATM define-se como um conjunto de acções realizadas pela rede durante o estabelecimento de um VC para determinar se uma conexão pode ou não ser aceite. Baseado nesta função, um pedido de estabelecimento de conexão é aceite apenas quando existem recursos suficientes em cada elemento da rede, de modo a garantir o contrato de tráfego mantendo a QoS das conexões já existentes.

A modelação de tráfego é um processo que altera as características do tráfego de um fluxo de células de uma conexão, com vista a garantir a conformidade com o contrato e assim obter uma melhor eficiência da rede mantendo a QoS definida. Alguns exemplos de modelação são a redução do débito de pico, a limitação do tamanho do *burst*, a redução do CDV através de um adequado espaçamento das células no tempo e através de políticas de escalonamento. A decisão de implementação ou não de um processo de modelação e o local onde é realizado é específico da rede. É ainda possível implementar funções de modelação nos sistemas finais para garantir que as células geradas pelas fontes estão em conformidade com os parâmetros de tráfego negociados.

2.1.4 Modelos de QoS em IP

Existem várias modelos de serviço para fornecer QoS em redes IP às aplicações. Destes modelos é possível destacar o *Integrated Services* (*IntServ*) [103, 78, 6] e o *Differentiated Service* (*DiffServ*) [32]. O modelo de serviço *IntServ* é orientado ao fluxo e usado na rede de acesso devido às suas características, enquanto que o modelo de serviço *DiffServ* foi idealizado para tratar fluxos agregados e por isso pode ser utilizado na rede de acesso e na rede de núcleo.

Integrated Services

O modelo de serviço *IntServ* tem como principal objectivo o fornecimento de QoS a aplicações com vários tipos de requisitos. Para conseguir esse objectivo este modelo usa um protocolo de reserva de recursos designado por *ReSerVation Protocol* (RSVP) [125]. O facto deste modelo ser orientado ao fluxo obriga que os *routers* mantenham informação de estado para cada fluxo, sendo o *router* que determina que fluxos obtêm recursos baseado na capacidade disponível.

O modelo *IntServ* permite tráfego com diferentes requisitos, por exemplo *Best Effort* (BE) para tráfego sem requisitos específicos, *Controlled Load* (CL) para tráfego do tipo BE, mas com controlo de congestionamento e o *Guaranteed Service* (GS) que foi desenvolvido para aplicações com requisitos de tempo real. Contudo, para fornecer estas diferentes garantias de QoS o tráfego é controlado através do *Controlo de admissão* que serve para verificar a existência de recursos suficientes para assegurar os novos pedidos, mas sem prejudicar os já existentes. Este modelo usa um *Classificador de pacotes* para

classificar os pacotes de acordo com a classe de tráfego escolhida e um *Escalonador* para distribuir a largura de banda pelos fluxos, de acordo com as características da classe de tráfego, para assegurar a sua equidade.

O protocolo de reserva de recursos associado (RSVP) tem como atributos essenciais a reserva que é iniciada pelo(s) receptor(es), é *soft-state*, e suporta *unicast* e *multicast*. Apesar da reserva ser iniciada pelo(s) receptor(es) é através de uma mensagem "*Path*" enviada pela fonte em direcção ao(s) destino(s) que recolhe a informação ao longo do caminho e que permite à chegada ao(s) destino(s) verificar, com base na informação recolhida, se a rede pode assegurar os recursos necessários, sendo de seguida gerada uma mensagem "*Resv*" solicitando a respectiva reserva dos recursos. Se os *routers* têm suficiente capacidade, então os recursos são reservados, senão é gerada uma mensagem de erro e enviada para o(s) destino(s). Como este protocolo é *soft-state* o estado da reserva é mantido através de um refrescamento regular.

A utilização do protocolo RSVP para reserva e sinalização levanta alguns problemas, nomeadamente a sua complexidade quanto à classificação de pacotes e ao escalonamento, a escalabilidade no número de destinos por fluxo (não escalável para muitos fluxos), o que o torna adequado para redes de pequenas dimensão mas não para grandes *backbones*, em que o número de fluxos pode ser muito elevado. Além deste problema, a não utilização da agregação de fluxos na rede de núcleo cria outro. Este problema é ainda mais complicado, porque obriga os *routers* ao longo do caminho estabelecido a manterem um registo dos requisitos e verificarem a conformidade dos pacotes.

Differentiated Services

Como alternativa ao modelo anterior foi proposto um mecanismo que possibilita uma diferenciação dos serviços de forma a ter em consideração a escalabilidade e, como o modelo anterior tratava os fluxos individuais, este novo modelo, designado por *DiffServ* [32] em vez de manter fluxos individuais em todos os *routers*, agrega fluxos que recebem tratamento por classe ou por serviço (designado por *Behavior Aggregate*, BA). As classes de serviço são identificadas através da marcação dos pacotes como pertencentes a um determinado serviço; os *routers* no caminho examinam o cabeçalho para determinar o tratamento a efectuar.

Basicamente, o *DiffServ* funciona com o IPv4 e com o IPv6. Quanto ao IPv4

utiliza alguns *bits* do cabeçalho antes previstos como *Type of Service* (ToS), mas que não estavam a ser utilizados. Este modelo fornece actualmente dois tipos de serviços: o *Expedited Forwarding* (EF) que garante um comportamento análogo ao de uma linha alugada e o *Assured Forwarding* (AF) para serviços que necessitam de mais garantias do que o BE.

Para assegurar o funcionamento descrito é necessário apresentar a forma como o modelo é constituído e quais as funções que deve realizar. Relativamente às funções existem quatro, nomeadamente

1. Controlo de admissão: capacidade da rede em recusar "clientes" quando o pedido excede a capacidade;

2. Escalonamento: método para tratar de forma diferenciada os pacotes;

3. Classificação do tráfego: capacidade de ordenar os fluxos em fluxos classificados para receberem um determinado tratamento;

4. Políticas e regras para a distribuição de recursos na rede.

Quando se observa o modelo *DiffServ* na perspectiva dos *routers* são definidas as classes e o respectivo tratamento a dar aos pacotes (designado por *Per-Hop Behavior* (PHB)). Os *routers* reservam recursos suficientes para cada classe e ordenam os pacotes que recebem na classe correspondente. Contudo, este modelo não necessita de armazenar informação complexa dentro dos *routers* sobre sessões individuais, mas apenas observar um único *byte* do cabeçalho do pacote antes de decidir qual o comportamento de QoS a aplicar, comportamento esse que está armazenado no *Differentiated Services Code Point* (DSCP).

Este modelo de QoS apresenta alguns problemas, dos quais se pode realçar a necessidade de ter vários PHBs por BA em cada nó, uma vez que é diferente estabelecer as garantias extremo-a-extremo quando comparado com o somatório de todos os PHBs, o que torna difícil de conceber serviços extremo-a-extremo com garantias nos nós individuais. A QoS é apenas para fluxos agregados, o que é útil, em especial para os ISPs e não para utilizadores individuais.

2.2 Evolução das arquitecturas IP/ATM

Num passado recente, emergiram várias soluções dentro do IETF e do ATM Forum, para suportar o protocolo IP sobre redes ATM, estando algumas adequadas para redes locais (LANs) e outras para grandes redes (WANs). Relativamente às LANs foi proposto o *LAN Emulation* (LANE) [40] e o *Multi-Protocol over ATM* (MPOA) [14, 15]. O MPOA tem como objectivo suportar outros protocolos da camada de rede para além do IP (IPX, Appletalk, etc.), com a utilização do LANE em conjunto com o *Next Hop Resolution Protocol* (NHRP) [76], de forma a permitir aos sistemas finais estabelecerem directamente VCs ATM. Outra das propostas para as LANs foi a utilização do *Classical IP over ATM* (CLIP) [74] entre *hosts* IP, em conjunto com o NHRP. Enquanto que para as WANs é usado o CLIP (entre *routers* IP) e o NHRP.

As soluções baseadas no "*Multilayer Switching*" (uma espécie de modelos híbridos) aparecem com o objectivo de tirar partido dos mecanismos de encaminhamento do IP e das técnicas de comutação do ATM. Estes esquemas permitem a utilização de IP e ATM em conjunto, com diferentes graus de compatibilidade, onde se podem destacar a utilização de *routers* baseados em células. Estes dispositivos são híbridos porque têm funcionalidades do IP (na camada 3) e funcionalidades do ATM (na camada 2). Basicamente, encaminham pacotes IP ao longo de VCs estabelecidos entre comutadores adjacentes; no entanto, é ainda possível estabelecer circuitos extremo-a-extremo e os pacotes serem encaminhados com base em células, permitindo assim libertar os equipamentos do interior da rede das funções necessárias ao encaminhamento de pacotes IP. Alguns dos construtores de equipamento de telecomunicações conceberam esquemas capazes de tirar partido das facilidades concedidas pelos algoritmos e mecanismos da camada 3, com a velocidade e fiabilidade fornecida pela comutação da camada 2, onde se destaca a Cisco com o *Tag Switching* [28], a Ipsilon com o *IP Switching* [88], a Toshiba com o *Cell Switch Router* [69], a IBM com o *Aggregate Router-Based IP Switching* (ARIS) [118] e, por fim, o IETF com o *MultiProtocol Label Switching* (MPLS) [99].

2.3 Tendência actual de integração

Nesta secção é apresentada e discutida a tendência actual na integração de várias tecnologias e a forma como essa integração é assegurada quando se pretende obter QoS extremo-

a-extremo e usar diferentes tecnologias. Em primeiro lugar são discutidos alguns dos problemas que se colocam relativamente à integração da tecnologia IP com a ATM. De seguida é apresentado o modelo geral para que a integração seja possível com o mínimo de alterações. Após o modelo estar definido são referenciadas algumas soluções para a tecnologia IP, enquanto que é analisado qual a melhor categoria de serviço a ser utilizada na rede ATM e por fim é discutida a forma como são suportados os modelos de serviço com QoS.

2.3.1 Problemas

Antes de discutir a integração das tecnologias é necessário observar um conjunto de questões que necessitam de ser clarificadas. Em primeiro lugar, quais são as aplicações alvo? Pergunta que não tem uma resposta muito fácil devido à constante evolução dos requisitos das aplicações nas últimas décadas. É possível identificar etapas em que determinadas aplicações emergiram, por exemplo, a execução remota de tarefas, serviços de FTP e de correio electrónico, mais recentemente tráfego *web* e actualmente e num futuro próximo tráfego multimédia; no entanto existe uma tecnologia que tem resistido, através do desenvolvimento de novos modelos de serviço (*IntServ* e *DiffServ*), a este aumento de exigências das aplicações que é o IP. Com base nesta situação a solução seria encontrar uma alternativa que permitisse realizar uma diferenciação dos vários tipos de serviços em redes IP, mas é necessário questionar sobre quais são, de facto, os verdadeiros requisitos que as aplicações têm nos dias de hoje.

Em segundo lugar é necessário discutir as seguintes questões: Que problemas é que se estão a resolver? Dar um serviço melhor a um determinado tráfego em detrimento de outro? Antes de responder é necessário definir o que se entende por serviço. O serviço pode ser "observado" de dois pontos de vista: do lado dos ISPs que desejam um controlo mais "apertado" relativamente à distribuição da largura de banda, em particular quando existe uma elavada carga na rede (implementação de níveis de eliminação, por exemplo), e do lado dos utilizadores que desejam ter reservas de recursos extremo-a-extremo, independentemente de outro tráfego (implementação de filas prioritárias e policiamento apertado).

Por último é necessário discutir o problema da escalabilidade das redes, devido ao crescimento vertiginoso que se tem observado nos últimos anos. Assim, um mecanismo

que possibilite uma diferenciação dos serviços deve ter em atenção o facto da escalabilidade na Internet, quer em termos de número de redes e utilizadores, quer em termos da heterogeneidade relativa às velocidades encontradas. Para obter esta escalabilidade, este tipo de mecanismos deve transferir toda, ou quase toda a informação de estado, para a entrada da rede e implementar mecanismos de policiamento e modelação na entrada, para garantir os diferentes serviços e deixar para a rede a tarefa de um transporte rápido.

2.3.2 Modelo de integração

A conjugação de uma tecnologia sem qualquer tipo de "depêndencia" (IP), com uma que permite uma maior velocidade de transferência e fornece garantias (por exemplo tecnologias de comutação de etiquetas, ATM ou MPLS), permite sobre a mesma suportar tráfego com qualquer tipo de QoS, possibilitando o fornecimento de garantias a aplicações IP com requisitos de QoS. A par destas qualidades conjugadas é necessário ter em consideração o índice de penetração da primeira no mercado (o exemplo mais paradigmático é a própria Internet), enquanto que a segunda é mais indicada para redes de núcleo.

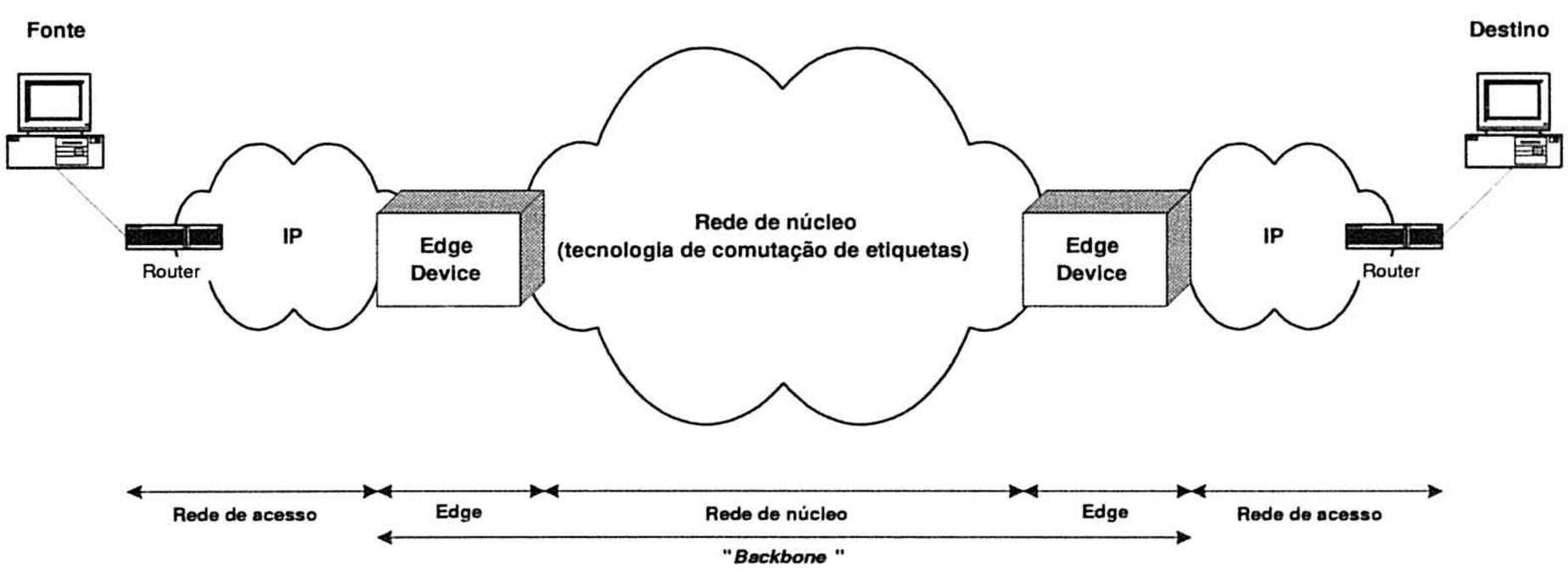

Figura 2.2: Estrutura geral do modelo.

O modelo apresentado em [46, 109], separação das várias redes, tem recebido a mesma atenção de outros autores [55] que veêm com agrado as vantagens da combinação dos dois tipos de dispositivos (ED e rede de núcleo) como, por exemplo, a arquitectura da Figura 2.2, que se baseia na separação em rede de acesso e *backbone* (o *backbone* é depois subdividido em ED e rede de núcleo), com vista a alcançar uma redução da complexidade, uma maior escalabilidade e um melhor desempenho. A necessidade dessa separação fica bem patente na seguinte afirmação "*(...) As a result, the fundamental principle of* "Leave complexity at the 'edges' and keep the network 'core' simple" *is a*

central theme among QoS architecture designs." [108].

Com base nas características dos dois tipos de dispositivos é possível criar cenários arquitectónicos que permitam tirar partido de ambos os dispositivos, conseguindo-se assim dar garantias de QoS às aplicações que usam o protocolo IP como, por exemplo, aplicações de voz e vídeo. Desta forma, e com base nas características de ambos, o modelo arquitectónico escolhido é a utilização do protocolo IP na rede de acesso e uma tecnologia de comutação de etiquetas (podendo ser o ATM ou provavelmente no futuro o MPLS que por sua vez pode usar o ATM como tecnologia de comutação) na rede de núcleo interligadas por sistemas periféricos (EDs).

2.3.3 Protocolos da rede de acesso

A escolha de um protocolo que suporta QoS sobre IP para funcionar na rede de acesso pode não ser tarefa fácil devido às características e filosofia dos protocolos existentes. Dos protocolos apresentados em [108] podem destacar-se os de reserva de recursos e os de atribuição de prioridades. Nos protocolos de reserva, os recursos da rede são reservados de acordo com os requisitos de QoS das aplicações e a largura de banda é sujeita a uma gestão rigorosa. Quanto aos protocolos de atribuição de prioridades o tráfego é classificado e os elementos da rede dão um tratamento preferencial aos pacotes classificados com maiores exigências de garantias.

Algumas das ideias que têm vindo a ser desenvolvidas por alguns dos autores assentam em dois tipos: (i) políticas de eliminação inteligente que permitem uma indicação implícita do débito, por exemplo o "*Early Packet Discard*" (EPD), o "*Partial Packet Discard*" (PPD) e o "*Random Early Detection*" (RED) e (ii) indicação explícita do débito, com alguns autores a defenderem a utilização de um *Acknowledgement Bucket* para regular o fluxo de tráfego TCP sobre ATM [85] (alteração da informação do cabeçalho dos pacotes ACK TCP com base no débito fornecido pela rede), enquanto que outros autores defendem a realização de alterações apenas na camada IP para permitir a utilização de aplicações TCP e UDP. Neste subconjunto encontra-se a proposta "*Explicit Congestion Notification (ECN) to IP*" [93] e algum trabalho desenvolvido por um grupo de investigadores do Massachusetts Institute of Technology (MIT), na concepção de uma nova arquitectura para a gestão do congestionamento com um protocolo para a gestão de tráfego extremo-a-extremo, com a indicação explícita do débito (processado pelo camada

IP) ao qual a fonte deve transmitir [2].

A comunicação entre fontes IP e os EDs aparece cada vez mais como um tópico de interesse, uma vez que é necessário tomar algumas decisões importantes, nomeadamente a alteração da camada IP nos *hosts*. Um dos trabalhos apresentado em [55] aborda esta questão e sugere a realização de alterações na camada IP, dado que a indicação implícita do débito levanta problemas de controlo devido aos diferentes RTTs das várias sessões TCP. A utilização da técnica de gerar pacotes IP quer no *host* quer no ED é útil porque permite solicitar ao ED alterações no débito, assim como outro tipo de informações.

2.3.4 Escolha da categoria de serviço para a rede de núcleo

A rede ATM com a sua capacidade de consolidar múltiplos serviços através da utilização de VPs consegue reduzir o número de conexões, permitindo o aumento da escalabilidade e do desempenho. A questão que se pode levantar neste momento é qual a categoria que deve ser utilizada para suportar fluxos de tráfego IP com requisitos de QoS. Segundo [100], é necessário utilizar VPs de dois tipos: *real-time* para mapear tráfego IP que necessite de garantias estritas e *non-real-time* para o outro tipo de tráfego. Neste caso serão analisadas apenas as categorias de serviço nrt-VBR e ABR com um débito mínimo garantido para tráfego de fluxos agregados.

É possível agora discutir os prós e os contras da escolha de uma determinada categoria de serviço para os VCs/VPs da rede de núcleo. Os VPs do tipo CBR têm como característica fundamental a sua natureza estática quanto à reserva da largura de banda. Os VCs dentro do VP vão sendo adicionados até a largura de banda se esgotar. Contudo, a natureza *burst* do tráfego da maior parte das aplicações IP não é bem acomodado em VPs estáticos porque estes restringem o acesso à largura de banda remanescente na rede de núcleo e, por outro lado, reduzem os ganhos da multiplexagem estatística.

Com o objectivo de escolher qual a melhor categoria foram estudados em [46] duas: CBR e nrt-VBR. Os resultados obtidos dependeram, por um lado, do tipo de tráfego que as aplicações geram e, por outro lado, as próprias características das categorias. A primeira (CBR) tem uma característica fundamental que é a garantia total do QoS, isto é, é estabelecido um VC no início e a rede garante as mesmas características até ao fim. Contudo, a sua pouca, ou nenhuma flexibilidade e, sobretudo, o tipo de

tráfego gerado levam à obtenção de resultados não muito satisfatórios. Da experiência realizada concluiu-se que o desempenho da categoria de serviço nrt-VBR consegue obter melhores resultados, relativamente ao número de *buffers* necessários nos elementos de ligação entre as duas tecnologias e à QoS obtida, dado que o nrt-VBR pode tirar melhor partido da sua capacidade de transmitir *bursts* com um débito instantâneo superior.

VCs nrt-VBR dentro de um VP nrt-VBR

A utilização da categoria de serviço nrt-VBR ao nível dos VPs e dos VCs apresenta uma vantagem inegável que é a garantia da largura de banda negociada durante o estabelecimento da conexão. Esta garantia é dada por parte da rede em relação ao tráfego garantido de pelo menos o valor contratado; valor reflectido pelo SCR, sendo ainda permitido a transmissão até ao valor máximo, também acordado no início, desigando por PCR. No entanto, apresenta uma desvantagem importante em relação à gestão dos recursos. A garantia de largura de banda dada pela rede a esta categoria é realizada em relação à banda contratada, mas não em relação à que pode ficar disponível, uma vez que o débito médio de longo prazo é sempre limitado por SCR.

A par da desvantagem anterior existe ainda a possibilidade de ser ineficiente quando os recursos são estimados ao valor do PCR e não do SCR, podendo assim ocorrer perdas na rede de núcleo [37, 100], a não ser que exista um mecanismo de marcação de acordo com o SCR, o que torna ineficiente e aumenta o atraso.

Como conclusão, a aplicação desta categoria de serviço em ambos os níveis não permitiria disputar a largura de banda remanescente em ambas as situações. Numa visão macroscópica, não existe qualquer "disputa" em relação à banda remanescente entre os vários VPs e depois entre os VCs pertencentes a um determinado VP. Desta forma é possível ter, por um lado largura de banda por ocupar ao nível dos VPs e depois dentro de um determinado VP entre os VCs e, por outro lado ter *buffers* de tamanho considerável para diminuir as perdas, para além da diminuição da escalabilidade.

VCs ABR dentro de VPs nrt-VBR

Com base na análise efectuada na subsecção anterior, pode-se dizer que esta opção apresenta mais vantagens do que a precedente. Em primeiro lugar, com base nas caraterísticas da categoria de serviço ABR, a largura de banda atribuída ao VP será

melhor gerida e utilizada do que na opção anterior. Em segundo lugar, devido à possibilidade de utilizar o MCR, é possível ter largura de banda garantida o que permite ter QoS e mesmo assim "disputar" a banda remanescente dentro do VP.

Uma das características do serviço ABR é a transferência do congestionamento para a periferia, onde é possível fazer a "*bufferização*" e, mais importante ainda é onde os fluxos são visíveis (antes de serem escalonados e agregados num determinado VC). Com o ABR existem poucas perdas na rede, além das aplicações tirarem vantagem de toda a largura de banda fornecida pela rede. Esta categoria de serviço fornece uma gestão de tráfego muito superior, dado que maximiza os *buffers* e a utilização da largura de banda devido ao conhecimento que os sistemas finais possuem relativamente à rede.

Como conclusão, esta opção apresenta melhores argumentos que a anterior, mas possui o mesmo problema ao nível dos VPs, porque não tem nenhum mecanismo que permita realizar uma distribuição da largura de banda remanescente pelos VPs que partilham uma mesma ligação.

VCs ABR dentro de VPs ABR

Esta combinação é a que apresenta o melhor comportamento quando discutida com base nas características da categoria de serviço ABR. Tanto ao nível de VC como ao nível do VP é sempre realizada uma distribuição equitativa da largura de banda pelas conexões em competição.

A utilização de canais ABR no suporte a fluxos IP agregados torna-se possível e viável, mesmo sabendo que esta classe não suporta na totalidade garantias quanto ao atraso mas, através da utilização de um débito mínimo garantido (total dos débitos médios agregados), de uma distribuição da largura de banda remanescente (obedecendo a um critério de *fairness*) e de um algoritmo de escalonamento é possível dar garantias quanto aos atrasos, às perdas e à gestão do *backlog*.

2.3.5 Suporte de serviços com QoS

Relativamente à garantia de QoS entre estas duas tecnologias existem diferenças assinaláveis, tais como a sinalização no IP para obtenção de QoS que é realizada à custa do protocolo de reserva de recursos RSVP para o *IntServ* ou o *DiffServ*, enquanto que no ATM é utilizado o *Private Network to Node Interface* (PNNI) e as categorias de serviço

(CBR, rt-VBR, nrt-VBR, ABR e UBR). Contudo, existem algumas similaridades que podem ser exploradas na interligação entre estas duas tecnologias, por exemplo, nos fluxos de tráfego é aplicado em ambos os modelos variedades de TBs para controlar e policiar o tráfego. Enquanto que no modelo *IntServ* são usados os parâmetros do TB, no ATM são usados os parâmetros PCR/CDVT e SCR/MBS. Relativamente às classes de tráfego suportadas pelo *IntServ* (GS, CL e BE) podem ser mapeadas em categorias de serviço do ATM (CBR, rt-VBR, nrt-VBR, ABR e UBR). Assim os serviços do tipo BE podem ser mapeados com VCs UBR, o CL pode ser mapeado em VCs nrt-VBR e o *GS*, que obriga a limites bem definidos quanto à largura de banda atribuída e ao atraso, pode ser mapeado em VCs CBR ou rt-VBR do ATM.

Alguns trabalhos foram realizados para tirar partido desta interligação e apresentam alguns resultados interessantes. O transporte de IP CL sobre ATM [46] mostra que, para aplicações com necessidade de garantias de serviço mesmo que seja transportadas sobre IP é necessário conceber mecanismos de *interface* entre as redes IP e as redes ATM que garantam os requisitos das aplicações; neste trabalho apenas foram usadas as categorias CBR e VBR. Devido às características da classe CL, os autores do trabalho [46] usaram um TB para realizar a modelação do tráfego e adequar aos parâmetros das categorias de serviço do ATM. Neste *interface*, os pacotes IP são segmentados e convertidos em células ATM, de acordo com o protocolo AAL5. Na rede ATM é definido uma conexão ATM, que será partilhada por todas as fontes interligadas ao mecanismo de *interface*.

O desempenho de ambas as categorias foi obtido com base num conjunto de métricas de QoS (percentagem de pacotes IP perdidos, atraso médio de transferência, ocupação do *buffer* ATM, percentagem de pedidos rejeitados e utilização da ligação ATM). Um dos resultados deste trabalho mostrou que a alteração dos valores dos parâmetros do TB provou alterações significativas nas métricas de QoS, em especial na capacidade do *buffer* ATM, mostrando que a categoria VBR apresenta vantagens sobre a CBR na transmissão de *bursts* com um débito de pico mais elevado. Como conclusão, o VBR explora melhor a natureza *bursty* deste tipo de fontes e, em ambos os casos, a ocupação da ligação ATM é semelhante; no entanto, a utilização da categoria ABR não foi explorada.

Quanto ao mapeamento do *DiffServ* no ATM não é fácil por causa do níveis de

eliminação de pacotes. Por exemplo, a classe AF fornece múltiplos níveis de eliminação e múltiplas classes, enquanto o ATM tem apenas dois níveis de eliminação através do *bit* CLP e um número não especificado de filas por cada categoria de serviço. Supondo que se tinha como objectivo mapear a classe AF numa rede ATM usando a categoria ABR, neste caso os EDs podem usar diferentes limiares de eliminação para os diferentes níveis de eliminação da classe AF, uma vez que a maior parte da filas estão no ED porque os fluxos são agregados num único VC ABR após o seu escalonamento. Relativamente às restantes categorias não é fácil tratar mais do que dois níveis de eliminação, porque as filas podem crescer dentro da rede e é difícil controlar os níveis de eliminação nesse ponto, devido ao facto dos fluxos não serem visíveis.

2.4 Questões relacionadas com conexões multiponto da rede ATM

A utilização de VCs do tipo multiponto (ponto-a-multiponto, multiponto-a-ponto e multiponto-a-multiponto) na rede de núcleo, apresenta um conjunto de características próprias relativas a cada tipo de conexão, levando ao aparecimento de problemas específicos. Nesta secção são discutidas as questões relacionadas com cada uma das conexões multiponto. No entanto é dado especial ênfase às conexões ponto-a-multiponto, pois a proposta apresentada no Capítulo 8 será referente a este tipo de conexão.

2.4.1 Ponto-a-multiponto

Os VCs do tipo ponto-a-multiponto são a primeira categoria de conexões multiponto que apresentam algumas vantagens sobre o modelo ponto-a-ponto apesar de levantarem algumas dificuldades que necessitam de ser ultrapassadas, têm, no entanto, uma particularidade em relação às restantes conexões multiponto, que é permitir a adaptação dos algoritmos de controlo de fluxo e de congestionamento ponto-a-ponto para este tipo de conexões.

A Figura 2.3 mostra a forma como a rede de núcleo pode ser configurada através da diminuição do número de VCs quando estes são do tipo ponto-a-multiponto. Este tipo de conexões permite tornar a rede de núcleo mais escalável, mas introduz problemas como o *feedback implosion* e o *consolidation noise*.

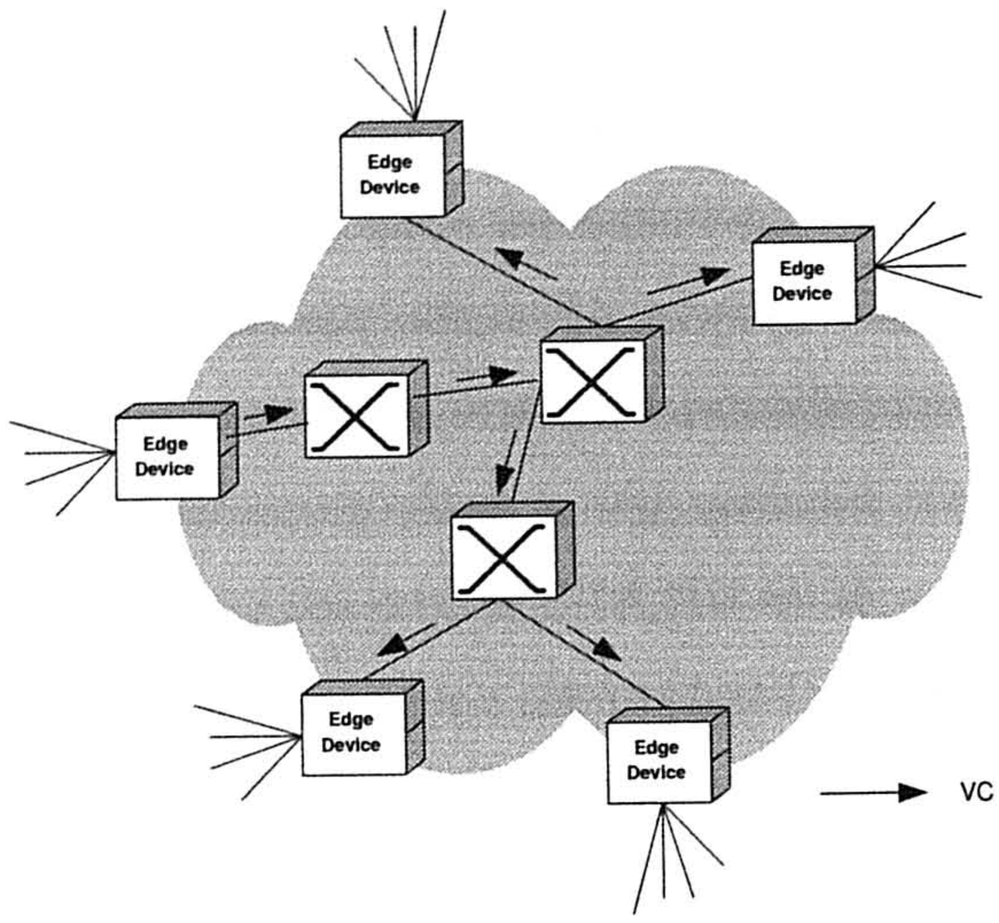

Figura 2.3: Arquitectura interna do *backbone*.

O *feedback implosion* pode ser entendido como a relação de células FRM geradas e o número de células BRM tratadas pela fonte. De forma simples percebe-se que, por exemplo, uma árvore *multicast* constituída por uma fonte e três destinos, se não existir qualquer tipo de controlo, a fonte por cada célula FRM que gerar vai ter de tratar três células BRM. Desta forma, basta um ligeiro crescimento da árvore para que este fenómeno tenha influência no desempenho global.

O segundo problema, conhecido por *feedback noise* ou *consolidation noise*, aparece quando os comutadores que têm funções de agregação não fazem a agregação total da informação constante das células BRM antes de enviar o resultado para montante, isto é, não esperam pela recepção de pelo menos uma célula BRM de cada um dos ramos de jusante que pertençam a uma determinada sessão. Os mecanismos que não têm em consideração este factor vão, inevitavelmente, sofrer do problema da inexactidão na informação que estão a passar para montante. Contudo, o problema não pode ser observado apenas nesta perspectiva, porque se o comutador estiver à espera até receber uma célula BRM de cada um dos ramos vai introduzir atrasos, que vão aumentado com o crescimento da árvore *multicast*, por isso, é necessário existir um compromisso entre a espera por todas as células BRM e o envio da informação para montante, como será ilustrado aquando da apresentação dos vários esquemas existentes, bem como na proposta apresentada.

2.4.2 Multiponto-a-ponto

As conexões multiponto-a-ponto estão relacionadas com a natureza do funcionamento dos protocolos de *routing* IP e portanto podem relacionar-se com as operações da rede de núcleo, o que não acontece para o tipo de conexões discutida anteriormente.

Os esquemas de controlo de fluxo deste tipo de conexões necessitam de alterações devido a dois problemas específicos. O primeiro problema é a impossibilidade de acontecer o *cell interleaving* quando o tipo de células usadas estão de acordo com o protocolo AAL5, o que implica ter comutadores que tenham a capacidade de resolver o *VC-merge*. O segundo problema existe porque as células provenientes das várias fontes necessitam de ser agregadas à medida que vão evoluindo em direcção ao destino, tendo como implicação directa a falta de informação sobre o que se está a passar para montante, o que vai obrigar a introduzir novos conceitos sobre a forma de distribuir a largura de banda pelos VCs.

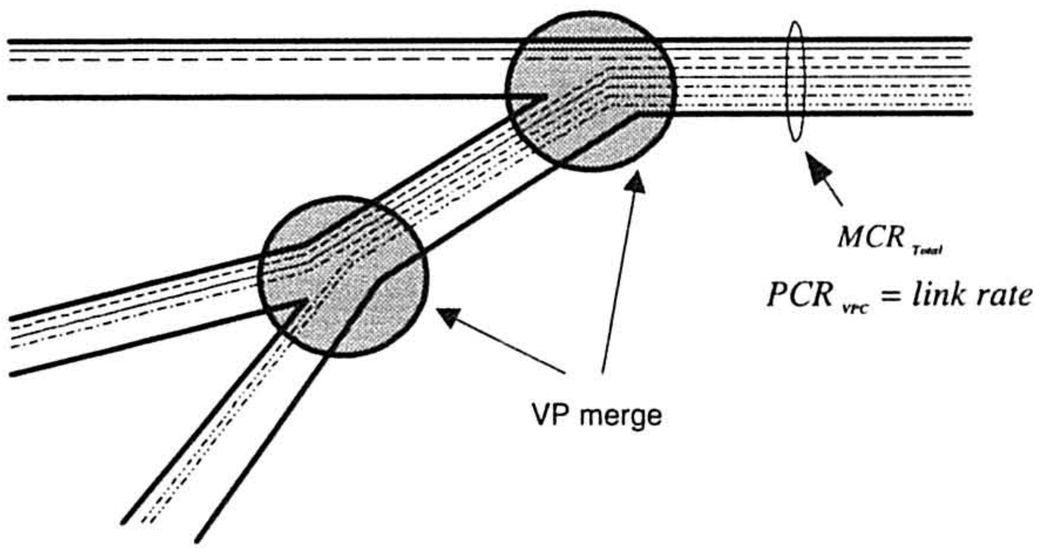

Figura 2.4: Indentificação de problemas em VPs do tipo multiponto-a-ponto.

Para resolver este fenómeno (*cell interleaving*) várias soluções têm surgido, onde as mais predominantes são baseadas na exploração da técnica do *VP-merge* e depois dentro de cada VP é possível ter um número de fontes a transmitir concorrentemente, dado que cada uma transmite no seu próprio VC (ver Figura 2.4). Além desta solução existem ainda autores que defendem a utilização de vários VCs em simultâneo, nomeadamente o *VC-pool* [31] e o *VC-bunching* [54], para além de algum trabalho já realizado na definição de regras e estratégias para conexões deste tipo [56].

2.4.3 Multiponto-a-multiponto

Os VCs do tipo multiponto-a-multiponto não são mais do que a combinação dos dois tipos anteriores e, por consequência, tem de se ter em consideração os problemas de cada

um deles. Apesar destas dificuldades existe já algum trabalho realizado, com destaque para o *Shared Many-to-Many ATM Reservation* (SMART) [45], o *Scalable and Efficient ATM Multicast* (SEAM) [52] e o método proposto por J. S. Turner [16, 114] que utiliza o campo *Generic Flow Control* (GFC) do cabeçalho das células ATM.

Como referenciado, neste tipo de conexões existem os problemas detectados nas conexões multiponto discutidas nas secções anteriores. Para resolver estes problemas é necessário muito trabalho, de modo a garantir que o congestionamento possa ser evitado em VCs deste tipo, independentemente da topologia, da localização do congestionamento, das características e dos padrões de tráfego. Qualquer uma das soluções apresenta vantagens e desvantagens, mas como este tipo de esquemas ainda se encontram numa fase "embrionária" é necessário mais trabalho para desenvolver e resolver os problemas conhecidos (detectados por cada um dos autores) e aqueles que virão a surgir em novos testes ou até em novas soluções.

Como será observado no capítulo 8, a grande preocupação será relativa aos problemas relacionados com a tecnologia ATM para conexões ponto-a-multiponto da categoria de serviço ABR, com especial ênfase na apresentação de uma proposta para a resolução dos problemas *feedback implosion* e *consolidation noise*.

Capítulo 3

Ferramentas de análise e simulação

Neste capítulo são apresentadas ferramentas matemáticas e de simulação que servirão para analisar as soluções obtidas para o escalonamento, para a distribuição da largura de banda dentro do ED e para os problemas específicos das conexões ponto-a-multiponto.

A necessidade de justificar (se possível formalmente) resultados obtidos por simulação levou à procura de ferramentas matemáticas aplicáveis à área, de modo a justificar as opções realizadas e consubstanciar os resultados alcançados.

Para além da ferramenta matemática recorreu-se a técnicas de simulação para a obtenção e verificação dos resultados sobre os algoritmos desenvolvidos. Estes resultados foram obtidos com base num simulador que foi modificado através da inclusão de novos componentes e dotando outros com novas funcionalidades. Foi ainda necessário definir cenários de forma a poder detectar todos, ou quase todos, os problemas que podem influenciar negativamente o desempenho das soluções encontradas.

Na primeira secção é apresentada uma ferramenta matemática de análise, enquanto que a segunda secção apresenta as propriedades fundamentais a considerar quando se decide construir um simulador. O simulador utilizado e os componentes adicionados são objecto da terceira secção e na última secção descrevem-se os cenários utilizados na obtenção de resultados.

3.1 Ferramenta matemática

Esta secção apresenta a ferramenta matemática *Network Calculus* com o objectivo de mostrar as suas potencialidades na aplicação ao modelo desenvolvido. Para uma ex-

plicação mais detalhada das operações, das definições e dos teoremas discutidos nesta secção, poderá ser consultado na secção A.1 do Anexo A.

Com o propósito de justificar teoricamente o modelo desenvolvido verificou-se que duas situações são interessantes do ponto de vista técnico e científico para justificar métodos de escalonamento com necessidades de garantias de serviço, nomeadamente as *curvas de serviço* introduzidas por Cruz [27] e a utilização de uma nova álgebra designada por *Max-Min plus* [12], onde o seu autor produz a seguinte afirmação "(...) *We show that all schedulers generally proposed for ATM or the Internet integrated services can be modeled by a family of simple service curves called the rate-latency service curves.*(...)". A par desta afirmação H. Sariowan, *et al* em [102] justificam da seguinte forma a utilização das *curvas de serviço* em modelos de escalonamento:

- A caracterização do serviço para uma conexão usando o conceito de *curvas de serviço* permite que a qualidade de serviço possa ser expressa como função simples de *curvas de serviço* e de *burstiness* do tráfego para cada conexão. Uma das primeiras definições está relacionada com o facto do somatório de todas as *curvas de serviço* não poder exceder a largura de banda disponível na ligação. Estas funções são independentes para cada uma das conexões e;

- Fornecem uma grande flexibilidade ao servidor na distribuição dos recursos de forma a obedecer aos requisitos de atraso e *throughput* para cada uma das conexões.

Com estas características é possível definir novas disciplinas de serviço com uma maior abrangência, mas sempre com uma grande dose de conservadorismo como discutido em [89]. É, no entanto, necessário verificar que a maior parte das políticas baseadas em débitos apenas fornecem limites inferiores relativamente ao *throughput*, o que resulta num desaproveitamento dos recursos quando uma conexão que tem tráfego do tipo *burst* (como é o caso das fontes de tráfego que estão a ser consideradas neste trabalho), e tem um requisito reduzido relativo ao atraso máximo, é obrigada a solicitar uma capacidade elevada para o seu *throughput* quando o débito médio é pequeno.

3.1.1 Operações

Na álgebra convencional as duas operações mais comuns dos elementos de $\mathcal{Z}$ e $\mathcal{R}$ são a adição e a multiplicação. Assim o conjunto de inteiros e reais envolvidos nestas duas o-

perações verificam um número de axiomas que definem as seguintes estruturas algébricas: $(\mathcal{Z}, +, \times)$ e $(\mathcal{R}, +, \times)$. No *Network Calculus* é considerada a álgebra *min-plus* em que as operações são alteradas da seguinte forma: a adição torna-se na operação de obtenção do mínimo e a multiplicação na operação de adição resultando na estrutura algébrica $(\mathcal{R} \cup \{+\infty\}, \wedge, +)$, com $\wedge$ a representar a operação de obtenção do ínfimo ou, se existir, o mínimo, tal que $a \wedge b = min\{a, b\}$ [12]. Esta nova estrutura algébrica, $(\mathcal{R} \cup \{+\infty\}, \wedge, +)$, em conjunto com a operação de convolução entre duas funções, designa-se por operação *min-plus convolution* (Definição 2) e resulta na função

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{f(t-s) + g(s)\},$$

se $t < 0$, $(f \otimes g)(t) = 0$.

A operação dual da operação anterior é a *min-plus deconvolution* (Definição 3); no entanto é alterada a operação de obtenção do inferior (ou mínimo) para a obtenção do superior, isto é, a notação $\vee$ representa o superior ou, se existir, o máximo, tal que $a \vee b = max\{a, b\}$, que é representada por

$$(f \oslash g)(t) = \sup_{u \geq 0} \{f(t+u) - g(u)\}.$$

Contrariamente à operação anterior, a função $(f \oslash g)(t)$ não é necessariamente zero se $t < 0$.

3.1.2 Funções de referência

Em [12] é apresentado um conjunto de funções de referência com as quais é "possível" modelar matematicamente outras curvas através das mais variadas combinações. Estas funções são crescentes em sentido lato se e só se $f(s) \leq f(t)$, para todo $s \leq t$. Das funções definidas é possível destacar aquelas que são utilizadas para a modelação dos resultados obtidos com os algoritmos desenvolvidos, nomeadamente a função *Rate-latency* $(\beta_{R,T})$ [1], *Peak rate* (λ_R) e a função *Affine* $(\gamma_{r,b})$; as funções encontram-se na Tabela A.1 e graficamente representadas na Figura A.1.

Com base nestas funções é possível fazer combinações com funções que representem o fluxo de chegada (quando relacionadas com as de referência) e conseguir obter funções de saída que respeitem os requisitos acordados, aquando do estabelecimento das

[1] É usada a notação $m^+ = max(m, 0)$

conexões. As curvas de base ou referência vão ser utilizadas em operações definidas mais adiante para obter novas curvas, no sentido de permitir caracterizar matematicamente os vários modelos de escalonamento, nomeadamente os modelos baseados na atribuição de tempos limite (*deadlines*).

3.1.3 Curvas de Chegada

O conceito de *curva de chegada* [102, 27, 12] é utilizado para modelar o tráfego de entrada e conseguir assim derivar os limites e a forma como o tráfego a partir desse momento está a circular na rede.

Nesta ferramenta foi estabelecido um conjunto de definições (4, 5, 6 e 8) e teoremas (7, 9 e 10) para caracterizar as *curvas de chegada*, e entre as mais utilizadas é possível encontrar a função *Affine*, $b(x) = \sigma + \rho x$, como a função que vai limitar superiormente o fluxo de entrada, A^{in}. Contudo, a escolha destas funções obriga à verificação de algumas condições, nomeadamente se a função é uma "boa" função, como referido na Definição 8.

3.1.4 Curvas de Serviço

As *curvas de serviço* introduzidas por L. R. Cruz em [27] e Le Boudec [12] são uma boa base para a justificação dos resultados deste tipo de algoritmos, nomeadamente as que, em primeiro lugar verificam as condições de "boa" função (Definição 8), e em segundo lugar obedecem às definições 11 e 12.

Ambos os autores [27, 12] afirmam que se existir um sistema e um fluxo a atravessar esse sistema, com as respectivas funções de entrada (A^{in}) e de saída (B^{out}) é possível definir uma *curva de serviço*, por exemplo, β, e garantir que o sistema oferece essa curva se $B^{out} \geq A^{in} \otimes \beta$, isto é,

$$B^{out}(t) \geq \inf_{s \leq t}(A^{in}(s) + \beta(t-s)).$$

Contudo, mesmo que o sistema apresente uma ocupação máxima, a *curva de serviço* definida anteriormente é sempre garantida.

3.1.5 Limites

Um dos objectivos dos autores de [27] e [12] quando estabeleceram as *curvas de chegada* e de *serviço* foi permitir estabelecer e derivar os limites de atraso e de *backlog*. Contudo, como foi já descrito, o objectivo deste trabalho não é derivar os limites máximos e mínimos de atrasos e *backlog*, mas a forma como estes são distribuídos.

A obtenção dos limites máximos relativamente aos atrasos e à ocupação dos *buffers* é dada pelos teoremas 13 e 16 para o primeiro caso (atraso), e pelos teoremas 14 e 15 para o segundo (*backlog*).

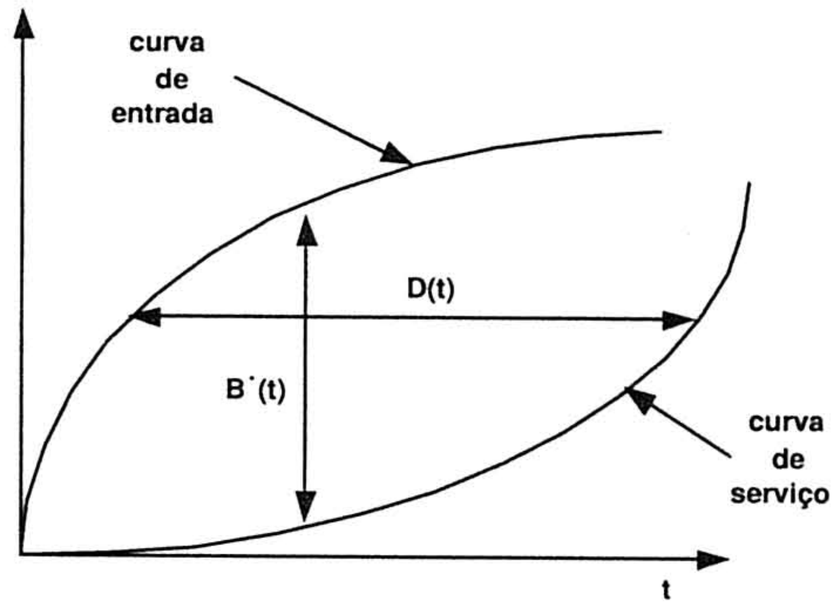

Figura 3.1: Exemplo de uma curva de chegada e de serviço.

Para perceber qual o significado gráfico destas definições e teoremas, a Figura 3.1 ilustra a *curva de serviço* $S(.)$ e uma *curva de chegada* $b(.)$. É possível observar que a máxima distância vertical entre as curvas $S(.)$ e $b(.)$ representa o *backlog*, $B^*(t)$, com

$$B^*[t] \leq \max_{s:s\geq 0}\{b(s) - S(s)\},$$

enquanto que a distância horizontal representa o atraso, $D(t)$, com

$$D[t] = \max_{s:s\geq 1} \min\{\Delta : \Delta \geq 0 \text{ e } b(s) \leq S(s+\Delta)\}.$$

Como será descrito no Capítulo 5 é possível justificar, baseado nesta ferramenta matemática, as escolhas efectuadas relativamente às *curvas de chegada* e de *serviço*, bem como a forma como os novos parâmetros introduzidos se relacionam com os limites definidos e possibilitam o estabelecimento de novas definições e teoremas.

3.2 Simulação

Uma simples pesquisa nos motores de busca da Internet da expressão "*Simulation*" ou "*Computing Simulation*" resulta numa quase infinidade de trabalhos das mais diversas áreas que recorrem à simulação para testar os seus modelos; dos resultados obtidos destacam-se apenas alguns para se verificar a diversidade de aplicações: "*Computer simulation of liquids*", "*The efficient simulation of parallel systems*", "*Plasma physics via computer simulation*", "*Computer simulation of hippocampal place cells*", "*Simulating the operations of the reusable shuttle space vehicle*", "*Simulation in industrial engineering and in problems of urban development (Mexico city)*", etc.

3.2.1 O porquê da simulação ?

As disciplinas de serviço ou modelos de controlo de fluxo e de congestionamento podem ser analisados usando várias técnicas, em que as mais poderosas do ponto de vista da obtenção de resultados rigorosos e confiáveis são a matemática e a simulação. Apesar da modelação e análise matemática ser bastante poderosa, não deixa de apresentar alguns problemas, tais como o conjunto de simplificações e pressupostos que necessitam de ser estabelecidos *à priori* mas que, no mundo real, e em especial na implementação em equipamentos, não são possíveis; por outro lado ignora interacções que podem ser críticas na prática, por exemplo, a perda de pacotes (ou células) pode desencadear um conjunto de retransmissões que podem causar novas perdas ou provocar o surgimento de outro tipo de fenómenos e assim sucessivamente. Por estas razões, a simulação não é apenas útil, mas necessária. No entanto, antes de construir um modelo de simulação é preciso, definir muito bem a que nível de detalhe se quer simular porque, dependendo da sua granularidade, a obtenção de resultados pode demorar minutos, horas ou até dias.

3.2.2 Decisões a ter em consideração na realização de experiências de simulação

Antes de se realizar um estudo de simulação é necessário observar um conjunto de propriedades importantes (nível de detalhe, que tipo de modelos de tráfego serão utilizados nas fontes, que tipo de topologia de rede e que métricas devem ser utilizadas para medir o desempenho) para a obtenção de resultados confiáveis.

Nível de detalhe

Inicialmente é necessário perceber a que nível de detalhe é que se quer realizar o estudo de simulação, por exemplo, ao nível físico, ao nível da rede, da camada de transporte, ao nível da célula, ou do fluxo, etc. Estas e outras questões devem ser previamente esclarecidas antes de tomar a decisão de construir um simulador, ou acrescentar módulos a simuladores já existentes. Como referido anteriormente, uma opção errada no nível de detalhe pode influenciar directamente o tempo que demora a obtenção dos resultados, bem como o tipo de resultados obtidos.

Neste trabalho foi utilizada uma ferramenta que simula fontes e comutadores baseados no controlo de células e pacotes, permitindo simular fontes com diferentes padrões de tráfego e aferir o desempenho de algoritmos de controlo de fluxo e de congestionamento e de disciplinas de serviço em relação aos atrasos verificados, à distribuição da largura de banda e à ocupação dos *buffers*.

Escolha das topologias de rede

Na escolha das topologias de rede é possível optar por topologias *benchmark* ou por topologias próprias para casos particulares, isto é, definir topologias de rede constituídas por fontes, equipamentos periféricos e comutadores com disciplinas de serviço e *buffers* com vários tamanhos com a sua interligação realizada através dos canais físicos (*links*) com a possibilidade de configurar vários débitos e distâncias.

As topologias de rede podem ser escolhidas para testar duas situações distintas: modelar uma rede existente e modelar o pior caso. A modelação de uma rede existente serve para isolar problemas e perceber as suas influências com o objectivo de obter novas soluções. A modelação do pior caso serve, como o nome indica, para prever e estudar o pior caso possível que possa vir a ocorrer, o que permite ter uma visibilidade mais clara sobre a "totalidade" dos problemas, permitindo soluções do tipo conservativo, mas que podem resultar em sobredimensionamento de recursos e portanto subutilização; no entanto, permite fixar limites superiores de parâmetros de desempenho.

Escolha das métricas

As três principais métricas de desempenho usadas no estudo das disciplinas de serviço são a distribuição da largura de banda, a distribuição do atraso e a ocupação dos *buffers*. A largura de banda pode ser entendida como o número de pacotes (ou células) enviados pela fonte para o destino num determinado período de tempo. O atraso de um pacote (ou célula) é o tempo necessário para percorrer um determinado percurso (ou parte do percurso) entre a fonte e o destino; para o caso em estudo é o intervalo de tempo que o pacote (ou célula) passa no equipamento periférico. A ocupação dos *buffers* refere-se ao número máximo de pacotes (ou células) que estão à espera de ser servidos dentro do equipamento periférico. Estas medidas permitem determinar se a disciplina de serviço apresenta equidade na distribuição da largura de banda e nos atrasos.

Nas simulações apresentadas estas medidas são obtidas sobre um determinado intervalo de tempo. Os atrasos verificados são medidos entre o instante de chegada (portas de entrada) ao equipamento periférico e o instante de saída ou fim de serviço (porta de saída), bem como a ocupação dos *buffers* e da largura de banda distribuída.

As métricas usadas na análise de desempenho das soluções para conexões ponto-a-ponto e ponto-a-multiponto são a eficiência, a *fairness*, o tempo necessário para o sistema atingir o estado estável e a sua adaptação a uma capacidade variável da largura de banda disponível e a fontes com vários modelos de tráfego. A realização dos testes de simulação baseou-se em configurações que tivessem diferentes atrasos de *feedback* (fontes localizadas a diferentes distâncias) e evidenciassem os problemas típicos deste tipo de conexões.

3.3 O simulador

O simulador utilizado foi inicialmente desenvolvido no MIT e melhorado qualitativa e quantitativamente pelo *National Institute of Standards and Technology* (NIST) [50] e fornece uma grande flexibilidade no estudo do desempenho das redes ATM. Esta ferramenta possibilita de forma interactiva modelizar o sistema através de uma *interface* gráfica. O desenvolvimento do simulador foi baseado na linguagem C e num sistema de janelas X (*Window System* X) para ser executado em plataformas Unix ou Linux.

A ferramenta permite ao utilizador criar diferentes topologias de rede, assim como

atribuir diferentes valores aos vários parâmetros dos componentes que compõem uma rede. Este simulador pode ser utilizado de duas formas distintas: (i) planear redes ATM ou (ii) analisar o desempenho de protocolos. Para planear uma rede ATM o utilizador pode usar o simulador para testar várias cargas na rede, assim como obter estatísticas do tipo de utilização das linhas, ocupação das filas de espera, etc. Alternativamente é dado ao utilizador a possibilidade de investigar os mecanismos de distribuição da largura de banda pelos diversos VCs, ou o *overhead* do protocolo, para além de ser possível implementar vários modelos (escalonamento ou controlo de fluxo e de congestionamento, etc.) e de seguida realizar estudos comparativos.

3.3.1 Componentes

O simulador é constituído por um conjunto de componentes e cada um por um conjunto de parâmetros configuráveis. Os componentes incluídos no simulador são: Aplicações ATM (AATM), Terminais de Banda Larga (BTE), Comutadores ATM (CATM) e Linhas Físicas (LF). Os CATM e os BTEs estão interligados por componentes LFs; no entanto, a ligação entre um AATM e um BTE não é considerada um componente. Estes componentes desempenham um conjunto de funções, que produzem resultados e que podem ser vizualizados graficamente na *interface* ou podem ser armazenados num ficheiro para posterior tratamento.

Linhas Físicas (LF)

Este componente simula o meio físico onde as células são transmitidas e dá a possibilidade ao utilizador de pode escolher o débito das linhas e o seu comprimento.

Comutadores ATM (CATM)

Este componente é usado para comutar/transferir células de cada VC entre duas LFs. Considerando que uma célula chega a um comutador proveniente de uma LF, no próximo *slot time* (tempo de célula), depois de algum atraso (definido pelo utilizador), o comutador pesquisa a tabela de encaminhamento (contém o número da rota, a próxima LF, o próximo CATM e o próximo BTE) para determinar a LF de saída. Nesta altura, se a LF tem um *slot* vazio o comutador coloca a célula na LF, se o *slot* não está disponível a célula é colocada numa das filas definidas, nomeadamente CBR/VBR, ABR ou UBR

dependendo do tipo de serviço definido para a conexão à qual pertence a célula. O comutador tem ainda disponível um mecanismo de controlo de fluxo para ABR e UBR e esquemas de gestão dos *buffers* em relação ao tráfego TCP/IP.

Dentro do comutador o utilizador pode manipular um conjunto de parâmetros: tempo de atraso, tamanho das filas de espera de saída, limiares associados às filas de espera, tipo de algoritmo de controlo de fluxo, etc., bem como implementar novos algoritmo e novo parâmetros.

BTE

É o componente que simula um terminal RDIS-BL (por exemplo, um computador) para além de poder ter uma ou mais AATM. Os parâmetros do BTE dependem do tipo de aplicação que a ele está associado; no entanto existem parâmetros comuns quando as AATM associadas são diferentes, como por exemplo, o tamanho dos *buffers* de entrada e de saída, etc. As células recebidas do lado da AATM são encaminhadas para a LF se existirem *slots* vazios, caso contrário são armazenadas numa das três filas definidas CBR/VBR, ABR ou UBR. O utilizador tem ainda a possibilidade de definir o tamanho máximo da fila para testar a percentagem de perdas e desta forma verificar a rapidez de recuperação dos modelos.

Aplicação ATM (AATM)

Este componente é o responsável por gerar tráfego que pode ser de dois tipos: constante ou variável. Para tráfego constante o simulador fornece uma aplicação do tipo CBR. Para o tráfego variável existem dois tipos de aplicações: *Poisson* e *Batch*. Nestas aplicações é possível definir o comprimento dos *bursts*, o comprimento do intervalo entre *bursts* e o débito de transmissão durante os *bursts*. As aplicações do tipo *Poisson* caracterizam-se por uma operação em dois períodos: um ON e um OFF; a duração do período ON (ou seja durante o *burst*) e do intervalo entre estes períodos são caracterizadas por uma distribuição exponencial. Durante o *burst* a aplicação transmite com um débito constante. Para tráfego de baixa prioridade o simulador fornece uma aplicação do tipo ABR que pode pertencer a duas classes: a primeira para conexões ponto-a-ponto e a segunda para conexões ponto-a-multiponto e ainda, dentro de cada classe, a três tipos: persistente (*greedy*), *Poisson* e *Batch*. Existe também a possibilidade de utilizar

aplicações do tipo TCP/IP. Por último existem mais duas aplicações para a categoria VBR: MPEG e *Self-Similar*.

3.3.2 Formato dos resultados

O simulador permite ao utilizador vizualizar graficamente a evolução do comportamento dos vários parâmetros e/ou guardar os valores num ficheiro histórico. O conjunto de gráficos fornecidos pelo simulador (Binário, Barra, Histórico e Histograma) permite adequar o tipo de gráfico ao tipo de informação fornecida pelos diferentes parâmetros. Os valores armazenados no ficheiro possibilitam um tratamento estatístico dos mesmos.

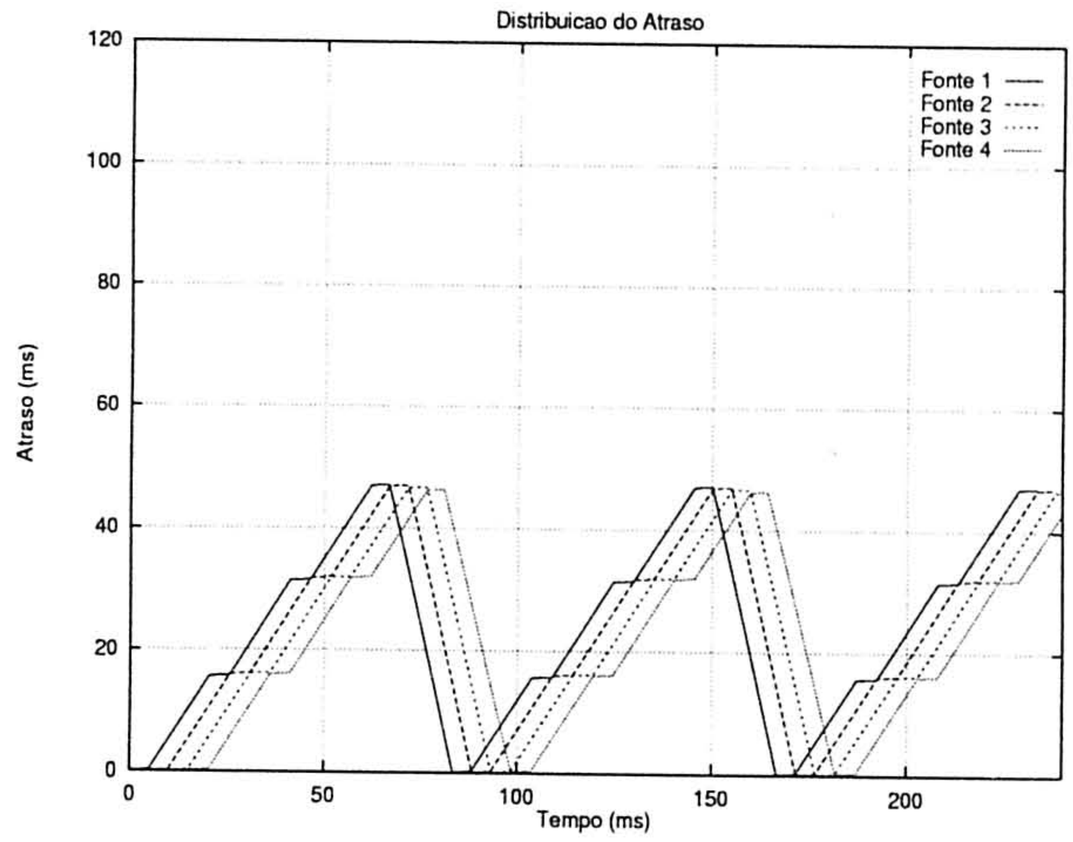

Figura 3.2: Exemplo de resultados.

O tipo de gráfico produzido com base nos valores obtidos após a utilização das aplicações referenciadas na secção **Aplicações adicionadas** poderá ser observado na Figura 3.2. Este gráfico serve apenas para ilustrar o tipo e o formato da informação utilizada na análise dos resultados das várias soluções.

3.3.3 Componentes e aplicações desenvolvidos

Para a obtenção dos resultados relativamente aos assuntos tratados neste trabalho foi necessário desenvolver componentes e aplicações específicos. Em relação às aplicações o problema a resolver era simples porque apenas era necessário tratar os valores armazenados pelo simulador ao longo da simulação. Quanto aos componentes a questão era mais complicada porque foi necessário desenvolver de raiz um componente que simulasse um equipamento periférico (*Edge Device*) com a capacidade de agregar fluxos através de um

esquema de escalonamento e mapeá-los num VC ATM. Além deste componente foi necessário dotar o simulador com capacidade de simular conexões multiponto de forma a ser possível testar os algoritmos de controlo e de consolidação dos VCs ponto-a-multiponto da categoria de serviço ABR.

Equipamento periférico (*Edge Device*)

O desenvolvimento de um componente no simulador em questão obriga a ter em consideração a forma como os componentes cooperam entre si, bem como as estruturas de dados associadas. No manual de *Utilização e Programação* do simulador [50] é apresentada a estrutura à qual os componentes devem obedecer para que seja possível a sua integração no simulador.

Com base na estrutura apresentada em [50] foi construído o componente *Edge Device* (apresentado com detalhe no Anexo A) com o objectivo de permitir simular algoritmos de escalonamento de tráfego IP a agregar em VCs ABR da rede ATM, bem como algoritmos de distribuição de largura de banda entre os vários VCs ABR em competição.

Capacidade de simular conexões multiponto

Apesar de não ter sido necessário desenvolver de raiz um componente específico para simular conexões multiponto, uma vez que as alterações a efectuar atravessaram praticamente todos os componentes (com excepção das linhas), foi necessário atribuir capacidade *multicast* para tratar o problema da consolidação das conexões ponto-a-multiponto da categoria de serviço ABR, para além da distribuição da largura de banda entre estas conexões e as ponto-a-ponto.

As alterações verificaram-se essencialmente no estabelecimento das rotas porque dentro dos comutadores, pertencentes a um VCs ponto-a-multiponto, era necessário enviar uma cópia de cada uma das células que chegavam de montante para os ramos a jusante e, em sentido contrário, era necessário realizar a consolidação das células BRM antes de enviar uma célula BRM em direcção à fonte.

Aplicações adicionadas

O simulador apresentava alguns problemas na fase de tratamento do ficheiro em que são armazenados os dados referentes aos parâmetros seleccionados. Um primeiro problema a resolver era a separação dos dados do ficheiro principal para ficheiros individuais que representassem cada um dos parâmetros. Um outro problema foi a necessidade de conversão dos valores obtidos para milisegundos para posterior representação. O primeiro problema foi resolvido através da construção de um pequeno *script* em Shell, uma vez que não exigia grande desempenho da plataforma utilizada, enquanto que para o segundo problema foi necessário construir uma pequena aplicação em linguagem C para efectuar alguns cálculos. Esta aplicação tem como função, além da conversão dos tempos para milisegundos, colocar os dados num formato de duas colunas (X, Y) que depois pode ser usado em qualquer programa que permita construir gáficos (Excel, GnuPlot, etc.). O programa usado na produção dos gráficos foi o GnuPlot porque é uma aplicação que oferece grande facilidade no manuseamento dos dados, bem como no tipo de gráficos que produz.

3.4 Cenários de simulação

Para a obtenção de resultados é necessário o estabelecimento de cenários de simulação de modo a que as soluções propostas possam ser devidamente avaliadas. A ideia de estabelecer configurações de base tem como objectivo avaliar as várias soluções e, como afirma Jain em [61], é preferível usar configurações simples, quando aplicáveis, porque permitem encontrar um grande número de problemas.

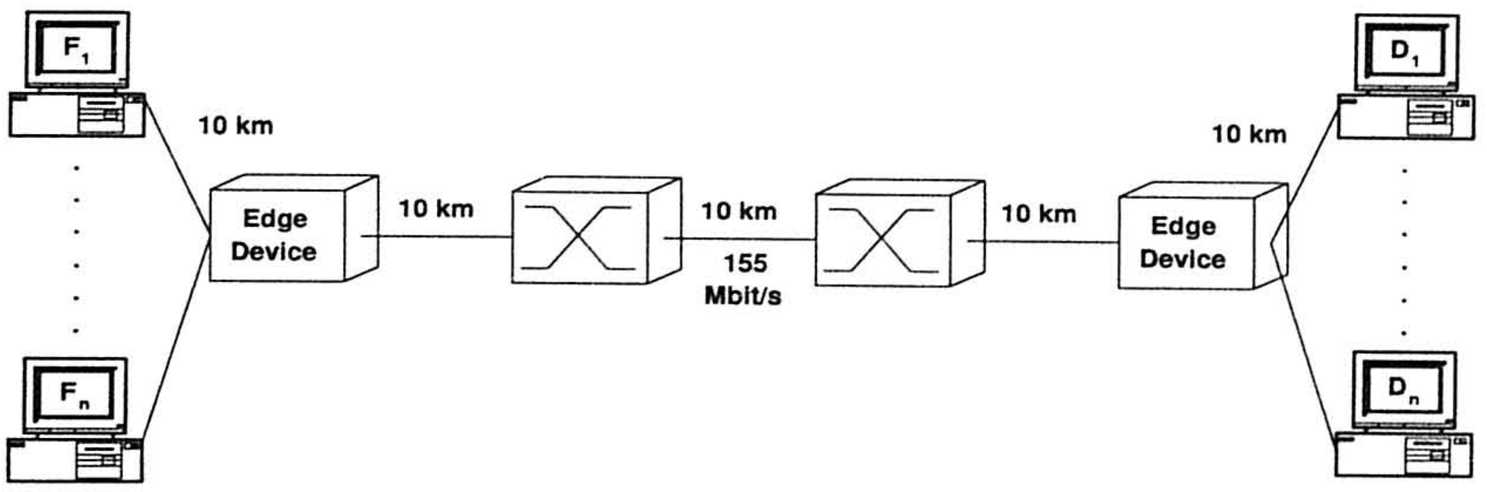

Figura 3.3: Configuração geral utilizada nos testes de simulação.

Com base na ideia defendida por Jain em [61] criaram-se seis cenários de si-

mulação sobre os quais serão testados os algoritmos de escalonamento. Estes cenários são caracterizados pela topologia de rede apresentada na Figura 3.3 constituída por N fontes do tipo IP, com o valor de N a variar entre 2 e 4. As fontes são caracterizadas por várias combinações dos parâmetros PCR, SCR e MBS (dessa combinação resultam valores para o tamanho do *bucket* (b), para o intervalo de tempo de transmissão do *burst* (T_{On}) e para o intervalo de tempo entre *bursts* (T_{Off})), em conjunto com a variação do grau de correlação temporal, com o objectivo de observar a forma como os algoritmos de escalonamento se adaptam às várias combinações. A avaliação dos "méritos" deste tipo de algoritmos é realizada com base nos diferentes cenários de tráfego, de acordo com critérios que incluem a distribuição dos débitos e dos atrasos, assim como no controlo da ocupação dos *buffers*.

Relativamente aos cenários para a simulação dos algoritmos de distribuição da largura de banda entre VCs ABR (ponto-a-ponto e ponto-a-multiponto) foram consideradas várias configurações *benchmark* [105], especialmente a configuração *Parking Lot* e duas configurações em *Cadeia*, utilizados pelo ATM Forum para avaliar os modelos deste tipo de conexões.

Capítulo 4

Trabalho relacionado

Com o enquadramento geral efectuado, os problemas caracterizados, o modelo de rede definido, as ferramentas de análise e de simulação apresentadas, é necessário analisar o trabalho efectuado nas áreas em estudo, de forma a perceber as vantagens e desvantagens das técnicas de escalonamento e de consolidação de células de gestão de recursos (RM) em conexões ponto-a-multiponto propostos na literatura.

Na primeira secção é estabelecida uma ordem na apresentação dos assuntos, bem como o modo como estes estão relacionados. Na segunda secção é apresentada uma classificação e descrição das disciplinas de serviço, bem como o seu modo de funcionamento. Na última secção são mostrados e discutidos os algoritmos de consolidação de células de gestão de recursos para conexões ponto-a-multiponto.

4.1 Introdução

Tomando como referência um modelo de rede constituído por uma rede de acesso e uma rede de núcleo interligadas por sistemas periféricos (*edge devices*) é necessário definir a forma como os pacotes IP, depois de serem segmentados em células, são escalonados e agregados em VCs da rede de núcleo.

Como já referido, as actuais aplicações têm requisitos de desempenho muito rígidos que exigem soluções tecnológicas capazes de fornecerem respostas adequadas. Uma das soluções para estas exigências é a tecnologia ATM que permite elevados débitos e, ao mesmo tempo, uma distribuição da largura de banda de acordo com as características de tráfego negociadas, sem esquecer outras tecnologias emergentes baseadas igualmente em comutação de etiquetas (*label switching*), como o MPLS.

A escolha de uma disciplina de serviço para o esclonamento de pacotes ou células dentro do ED necessita de obedecer a um conjunto de requisitos de forma a satisfazer os objectivos inicialmente definidos. Para alcançar esses objectivos é necessário em primeiro lugar verificar se alguma solução apresentada na literatura o consegue. Com este propósito será apresentado e discutido um número significativo de disciplinas de serviço para verificar se alguma consegue uma distribuição equitativa da largura de banda e do atraso.

Após a apresentação das várias propostas para os mecanismos a implementar no ED são ainda apresentados e discutidos mecanismos de consolidação de células de recursos (RM) em conexões ponto-a-multiponto na rede de núcleo para a categoria de serviço ABR.

4.2 Técnicas de escalonamento

Um dos problemas mais importantes no fornecimento da QoS é a escolha da disciplina de serviço, que determina a forma como os pacotes de diferentes conexões (fluxos) acedem a um recurso partilhado. Um outro problema que se prende com o tipo de tráfego que chega ao comutador ou aos dispositivos periféricos[1] é a sua caracterização; o tráfego das redes de pacotes e da maior parte das aplicações é do tipo *bursty*. Para este tipo de tráfego é necessário estabelecer parâmetros de desempenho para que seja possível a disciplina de serviço ter métricas para aferir o seu desempenho. Para conseguir isso é necessário definir os parâmetros de desempenho e as características do tráfego. Relativamente aos parâmetros de desempenho, H. Zhang em [122] apresenta quatro: (i) o limite de atraso extremo-a-extremo; (ii) a garantia de largura de banda; (iii) a variação do atraso extremo-a-extremo e (iv) a taxa de perdas. Quanto às características do tráfego estas foram já apresentadas e discutidas no Capítulo 2.

Após a definição dos parâmetros de desempenho é necessário indicar quais os requisitos para conceber uma disciplina de serviço. Em termos genéricos os algoritmos de escalonamento devem obedecer ao seguinte conjunto de requisitos: (i) Facilidade de implementação (tomadas de decisão rápidas e não depender se possível do número de conexões activas), (ii) Equidade dentro de cada classe (satisfazer o critério *max-*

[1] No caso de redes com tecnologias diferentes, por exemplo, interligação entre redes IP e redes ATM.

min fairness [18]), (iii) Limites de desempenho (a forma de obter o nível de serviço desejado) e (iv) Controlo de admissão (controlo da utilização dos recursos, por exemplo, um escalonamento eficiente pode influenciar a estratégia de admissão).

A partir do momento em que os requisitos de desenvolvimento de uma disciplina de serviço estão determinados é necessário discutir questões, tais como o número de níveis de prioridade vai ter (são servidos todos os pacotes de uma prioridade e só depois se passa à próxima, ou os níveis mais elevados fornecem limites de atraso mais apertados, etc.), se os escalonadores transmitem sempre um pacote assim que exista algum em espera ou podem deixar a ligação vazia mesmo existindo pacotes para servir (*Work-conserving vs Non-Work-conserving*), o grau de agregação (é possível agregar fluxos por classes com os mesmos requisitos de desempenho) e por fim o tipo de serviço dentro de um nível de prioridade (servido pela ordem de chegada ou servido de acordo com tempos de saída calculados à chegada). É ainda necessário definir se a disciplina de serviço tem por objectivo escalonar tráfego do tipo *best-effort* ou do tipo serviços garantidos porque os requisitos são muito diferentes, em que o *fairness* é o principal objectivo para o primeiro, enquanto que para o segundo são os atrasos e a largura de banda.

4.2.1 Disciplina de serviço FIFO

A forma mais simples de gerir e escalonar pacotes é manter filas do tipo FIFO em cada porta de entrada. Para escalonar pacotes o algoritmo examina os pacotes que estão à cabeça de cada uma das filas. O problema que se coloca numa situação destas é a existência de duas ou mais filas com pacotes com a mesma porta de saída como destino, o que vai obrigar o algoritmo de escalonamento a escolher um e um só pacote. Uma das possibilidades para resolver este problema é manter uma prioridade fixa, por exemplo, seleccionar a porta de entrada com o número mais baixo; no entanto é necessário perceber que esta solução é simples, fácil de implementar, mas injusta, uma vez que pode provocar "*starvation*". Outra das soluções seria uma escolha aleatória da porta de entrada a ser servida, seria mais justa mas também levaria a um aumento na complexidade de implementação.

Antes de apresentar as disciplinas de serviço com maior complexidade é necessário referir alguns modelos que têm apenas a preocupação em realizar o escalonamento em FIFO puro. De entre as várias soluções podem-se destacar as seguintes: *Cyclic* (ou

Round-Robin) selection [1]; *Global FIFO (ou eraliest arrival) selection* [1]; *HOL FIFO (ou oldest HOL) selection* [1, 21]; *Longest queue selection* (*Longest Queue First*, LQF) [1], *Oldest queue selection* (fila que está há mais tempo sem ser servida) [1] e *Random selection* em que, de entre todos os pacotes que estejam à cabeça das suas filas é escolhido aleatoriamente um deles [21]. Como se pode verificar existe um grande número de soluções "básicas" no escalonamento quando está presente o FIFO puro e quando não é necessário dar garantias de serviço, relativamente ao controlo dos atrasos, da distribuição da largura de banda e da percentagem de pacotes perdidos; estas preocupações serão objecto de discussão nas subsecções seguintes.

4.2.2 Classificação das disciplinas de serviço

Na última década foram propostos inúmeros modelos que permitem alcançar as garantias de QoS necessárias, como pode ser observado na literatura da área. Nesta secção serão apresentados alguns dos esquemas mais significativos das classes *Work-conserving* (WCing) e *Non-Work-conserving* (N-WCing).

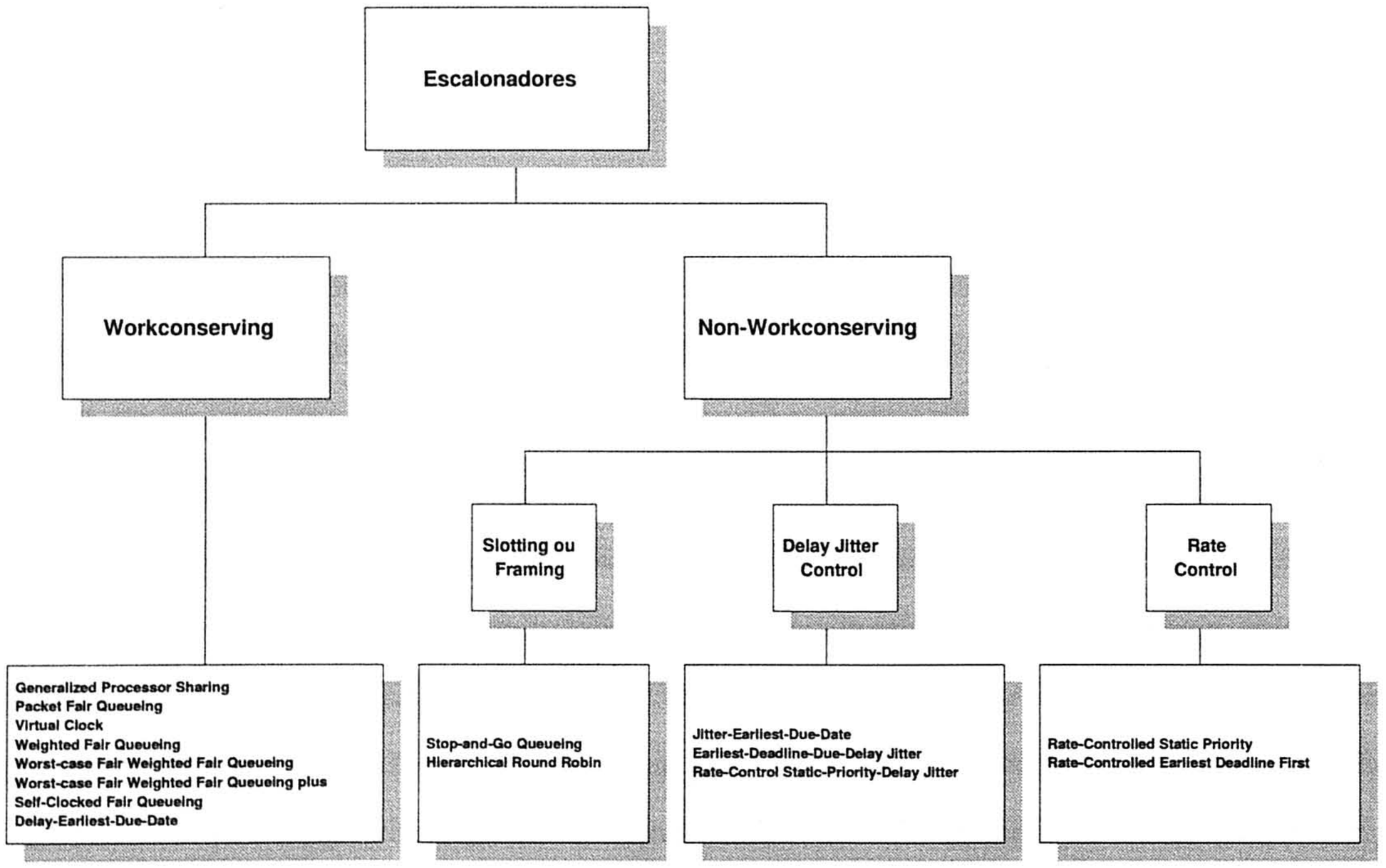

Figura 4.1: Classificação dos escalonadores de pacotes.

A Figura 4.1 ilustra a forma como os esquemas podem ser classificados de acordo com as suas características específicas. Os escalonadores WCing transmitem um pacote

sempre que exista algum em espera, enquanto que os escalonadores pertencentes à classe N-WCing podem em certas circustâncias deixar a ligação de saída vazia mesmo se existirem pacotes para transmitir.

Na classe WCing encontram-se classificados os seguintes mecanismos: *Generalized Processor Sharing* (GPS) [91], *Packet Fair Queueing* (PFQ), *Virtual Clock* [124], *Weighted Fair Queueing* (WFQ) [30], *Worst-case Fair Weighted Fair Queueing* (WF^2Q) [4], *Worst-case Fair Weighted Fair Queueing plus* (WF^2Q+) [5], *Self-Clocked Fair Queueing* (SCFQ) [48] e *Delay-Earliest-Due-Date* (Delay-EDD) [38].

A adopção de escalonadores N-WCing deve-se ao facto de nas disciplinas da classe anterior (WCing) o padrão de tráfego poder ser distorcido dentro da rede devido à variação da largura de banda existente em cada uma das ligações e por essa razão é complicado perceber que tipo de tráfego segue num determinado fluxo. Os escalonadores N-WCing aparecem como uma das soluções para a limitação apresentada, uma vez que utilizam um mecanismo regulador de débito antes do escalonamento propriamente dito, o que em contrapartida pode originar um aumento do atraso.

A ideia de deixar a ligação de saída vazia (classe N-WCing) está relacionada com o tipo de soluções encontradas, tais como:

1. *Slotting* ou *framing:* nesta subclasse de escalonadores encontram-se o *Stop-and-Go Queueing* [49] e o *Hierarchical Round Robin* [66]. Este tipo de modelos funcionam de forma análoga aos *time-division multiplexers* (TDM), em que o tempo é divido em *slots* e cada um dos *slots* é atribuído a um fluxo. Se no momento em que um determinado fluxo deve ser servido não existir nenhum pacote para transmitir, a transmissão de um pacote só poderá ocorrer no período seguinte.

2. Controlo da variação do atraso (*Delay jitter control*): a variação do atraso de um fluxo pode ser entendido como a variação do atraso extremo-a-extremo dos pacotes. Neste segundo grupo é possível apontar as disciplinas *Jitter-Earliest-Due-Date* [117] e *Earliest-Deadline-Due-Delay Jitter* [117]. A técnica utilizada por estas disciplinas de serviço para impor o limite na variação do atraso é o uso de *buffers* para manter os pacotes até que o atraso seja satisfeito.

3. Controlo do débito (*Rate control*): Se um escalonador tem um controlador de débito, o mecanismo monitora as chegadas e força cada fluxo ao acordo estabelecido

aquando do estabelecimento da ligação. Este grupo de escalonadores utiliza um controlador de débito que força que o tráfego de um fluxo seja encaminhado para o escalonador de acordo com as características negociadas inicialmente, podendo-se destacar o *Rate-Controlled Static Priority* [121] e o *Rate-Control Static-Priority Queueing* [123].

De seguida serão apresentados de forma resumida algumas disciplinas de serviço representantes de cada uma das classes (WCing e N-WCing).

4.2.3 Disciplinas de serviço Workconserving

Fluid Fair Queueing (FFQ) ou Generalized Processor Sharing (GPS)

O servidor FFQ ou (GPS) [91] é um servidor ideal, mas impraticável porque assume que o servidor pode servir todas as conexões com células, ou pacotes simultaneamente e que o tráfego é infinitamente divisível. Com base nos pressupostos anteriores e na realidade, verifica-se que é impossível de implementar, porque num sistema tradicional de pacotes, apenas uma conexão de cada vez pode receber o serviço do escalonador.

Generalizando o conceito, o servidor GPS serve N filas não vazias com um débito fixo, R - débito disponível na ligação de saída - e é caracterizado por N números reais positivos, $\phi_1, \phi_2, ..., \phi_N$, em que cada um corresponde a uma fila, com ϕ_i a representar a fracção de largura de banda atribuída ao fluxo i. Se $W_i(t_1, t_2)$ representar o montante de tráfego da sessão i servido no intervalo de tempo $[t_1, t_2]$, o servidor GPS é definido da seguinte forma

$$\frac{W_i(t_1, t_2)}{W_j(t_1, t_2)} \geq \frac{\phi_i}{\phi_j} \qquad j = 1, 2, ..., N \tag{4.1}$$

Esta condição verifica-se para qualquer fila i que tenha pacotes para serem servidos durante o intervalo $[t_1, t_2]$. Com base nesta definição é fácil observar que se $B(\tau)$ representar o conjunto de filas com pacotes para servir no instante de tempo τ, as restantes filas mantêm-se inalteráveis durante qualquer intervalo de tempo $[t_1, t_2]$, logo o débito de serviço da sessão i durante este intervalo de tempo é

$$r_i^*(t_1, t_2) = \frac{\phi_i}{\sum_{j \in B(\tau)} \phi_j} R \tag{4.2}$$

Como $B(t_1)$ é um subconjunto de todas as filas, logo

$$r_i^*(t_1, t_2) \geq r_i \tag{4.3}$$

e

$$r_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} R \tag{4.4}$$

desta forma a fila i tem um débito garantido de r_i durante o intervalo de tempo em que tiver células, ou pacotes para transmitir.

Packet Fair Queueing (PFQ)

Os algoritmos da classe de disciplinas de serviço conhecidos por PFQ têm recebido muita atenção porque são uma aproximação ao modelo ideal (GPS) que provou que tem duas proriedades desejáveis: (i) pode garantir um atraso extremo-a-extremo independentemente do comportamento das outras sessões e (ii) pode garantir, instantaneamente uma distribuição justa da largura de banda disponível pelas sessões com tráfego para transmitir. A primeira propriedade é a base para suportar tráfego com necessidades de garantias de serviço, segundo [91], enquanto que a segunda propriedade é importante para suportar tráfego do tipo *best-effort* [70].

As principais dificuldades destes algoritmos, como apresentado em [110], é a necessidade de *bufferização* por sessão e qual a arbitragem entre as sessões. Segundo os mesmos autores [110] existem três grandes custos associados com esta arbitragem: (1) o cálculo da função de tempo virtual[2] do sistema; (2) a gestão da fila com prioridades para ordenar a transmissão dos pacotes e (3) a gestão de uma segunda fila com prioridade para regular os pacotes.

Todos os algoritmos do tipo PFQ usam um mecanismo similar ao da fila com prioridade baseada na noção da função de tempo virtual; no entanto a distinção entre eles é realizada em dois aspectos: a função de tempo virtual usada e a política de selecção de pacotes, ou células. Para aproximar o GPS, um algoritmo do tipo PFQ mantém uma função de tempo virtual $V(.)$, o tempo de chegada $S_i(.)$ e o tempo de partida $F_i(.)$ para

[2] A função de tempo virtual representa uma medida dinâmica do montante de serviço que cada uma das sessões deve receber.

cada pacote da sessão i. As funções $S_i(.)$ e $F_i(.)$ são actualizadas de cada vez que a sessão i se tornar activa ou um pacote da mesma sessão terminar o seu serviço:

$$S_i(t) = \begin{cases} \max(V(t), F_i(t-)), \text{ sessão } i \text{ tornou-se activa} \\ F_i(t-), \ p_i^{k-1} \text{ terminou o serviço} \end{cases} \tag{4.5}$$

$$F_i(t) = S_i(t) + \frac{L_i^k}{r_i} \tag{4.6}$$

em que $F_i(t-)$ é o tempo virtual de partida do pacote anterior (p_i^{k-1}) pertencente à sessão i antes da actualização e L_i^k é o tamanho do pacote k que está à cabeça da fila da sessão i. Como pode ser observado, $V(t)$ representa o montante de tempo de serviço atribuído a cada sessão no instante de tempo t e $S_i(t)$ representa o montante de tempo de serviço recebido pela sessão i no instante de tempo t (ou tempo de chegada); logo o objectivo de todos os algoritmos do tipo PFQ é minimizar a discrepância que existe entre estas duas grandezas.

Virtual Clock

Este modelo tem como objectivo emular um sistema do tipo TDM, em que cada pacote tem atribuído um tempo virtual de transmissão, que é o instante de tempo em que o pacote seria servido (ou escalonado) se o servidor fosse do tipo TDM. O algoritmo *Virtual Clock* atribui a cada pacote um tempo de transmissão virtual baseado no padrão de chegada e na reserva da largura de banda à qual o pacote pertence, e é obtido da seguinte forma

$$auxVC_{i,j}^k \leftarrow max\{a_{i,j}^k, auxVC_{i,j}^k\} + Vtick_{i,j} \tag{4.7}$$

em que $auxVC_{i,j}^k$ é uma variável auxiliar para guardar o tempo virtual obtido para o pacote k, do fluxo i no equipamento j, $a_{i,j}^k$ é o tempo de chegada do pacote e $Vticks_{i,j}$ é o tempo médio entre chegadas de pacotes. Os pacotes são ordenados pelos seus tempos virtuais de transmissão e servidos por essa ordem.

Weighted Fair Queueing (WFQ)

A disciplina de serviço WFQ tenta aproximar o modelo ideal (FFQ ou GPS) porque quando o servidor está pronto para transmitir o próximo pacote no instante de tempo τ

ele vai escolher de entre os pacotes enfileirados no sistema nesse instante o primeiro que completaria o serviço no sistema GPS se nenhum pacote chegasse depois do instante τ. O algoritmo WFQ atribui a cada pacote um tempo virtual de transmissão com base na largura de banda reservada e é calculado da seguinte forma

$$F_{i,j}^{k} \leftarrow max\{V_i(a_{i,j}^{k}), F_{i,j}^{k-1}\} + \frac{L_j^k}{\phi_{i,j}} \tag{4.8}$$

em que $V_i(a_{i,j}^{k})$ é o tempo de chegada do pacote k, $F_{i,j}^{k-1}$ é o tempo de partida do pacote anterior, L_j^k o tamanho do pacote e $\phi_{i,j}$ a fracção da largura de banda atribuída. Os pacotes são ordenados pelos seus tempos virtuais de transmissão ($F_{i,j}^{k}$) e servidos por essa ordem.

A implementação desta disciplina de serviço também se apresenta complexa apesar dos excelentes resultados teóricos conseguidos. Para uma implementação aceitável foi utilizado a disciplina de serviço *Self-Clocked Fair Queueing* que será apresentada mais à frente.

Worst-case Fair Weighted Fair Queueing (WF^2Q)

Este modelo é uma evolução do anterior e utiliza os tempos de chegada e de partida de um pacote enquanto que o WFQ utiliza apenas o de partida. No WF^2Q quando o próximo pacote é escolhido para ser servido no instante de tempo τ, em vez de seleccionar o pacote entre todos os pacotes, o servidor apenas considera o conjunto de pacotes que tenham começado (e possivelmente terminado) o respectivo serviço no correspondente sistema GPS no instante de tempo τ, ou formalmente, $\{p_i^k | b_{i,GPS}^{k} \leq \tau \leq b_{i,WFQ}^{k}\}$ e selecciona entre esses o pacote que completaria o serviço em primeiro lugar no sistema GPS correspondente.

Como observado em [4] para fontes que medem a disponibilidade de largura de banda da ligação com base na recepção de informação da rede, é possível uma estimativa mais apurada e num curto intervalo de tempo quando os comutadores usam o WF^2Q em vez do WFQ. Para provar a eficiência deste modelo, os autores [4] utilizaram um servidor com débito controlado que é constituído por dois componentes: um conjunto de reguladores e um escalonador. Os pacotes ao chegarem ao comutador são tratados nos reguladores até os seus tempos elegíveis (o cálculo dos tempos elegíveis é realizado

de forma idêntica ao da disciplina anterior, isto é, $F_{i,j}^k \leftarrow max\{V_i(a_{i,j}^k), F_{i,j}^{k-1}\} + \frac{L_j^k}{\phi_{i,j}}$) antes de serem passados para o escalonador e, de seguida, o escalonador apenas serve esse conjunto de pacotes.

Worst-case Fair Weighted Fair Queueing Plus

Os autores que propuseram a disciplina de serviço WF^2Q apresentaram uma nova versão deste modelo conhecida por WF^2Q+ [5] em que o tempo virtual é actualizado apenas antes do pacote ser entregue. Neste instante o tempo virtual é incrementado com o tamanho do pacote dividido pela capacidade do servidor. Quando um pacote chega a uma sessão vazia o seu tempo virtual de início é igual ao tempo virtual do seu tempo de chegada. Por outro lado se a sessão não está vazia, o seu tempo de chegada é igual ao tempo virtual de saída do pacote anterior. Da mesma forma, o tempo virtual de saída de um pacote é igual ao tempo virtual de chegada mais o seu tamanho dividido pela largura de banda pré-atribuída ao fluxo ao qual o pacote pertence. Este algoritmo tem duas propriedades: (i) fornece o limite de atraso mais apertado entre todos os algoritmos desta classe e (ii) por analogia com o WF^2Q os pacotes escolhidos são os que apresentam um tempo de início inferior ao tempo virtual.

Self-Clocked Fair Queueing

Como discutido nos três últimos modelos, a manutenção do FFQ é complexa do ponto de vista computacional e com o propósito de aligeirar essa complexidade foi desenvolvida uma aproximação conhecida por "*Chuck's Approximation*". Nesta aproximação o tempo virtual do servidor é substituído pelo *virtual finish time* do pacote em serviço

$$F_{i,j}^k \leftarrow max\{\hat{V_i}(a_{i,j}^k), F_{i,j}^{k-1}\} + \frac{L_j^k}{\phi_{i,j}} \tag{4.9}$$

em que $\hat{V_i}(a_{i,j}^k)$ é o tempo de chegada do pacote, $F_{i,j}^{k-1}$ o último tempo de partida calculado, L_j^k o tamanho do pacote e $\phi_{i,j}$ a fracção da largura de banda atribuída.

O algoritmo é baseado na observação do tempo virtual do sistema que em qualquer instante de tempo t pode ser estimado a partir do tempo virtual de serviço do pacote que está a ser servido. A função de tempo virtual é definida por $F_{i,j}^k$ em que $s^k < t \leq f^k$, sendo s^k e f^k o tempo de início e fim do pacote k no sistema SCFQ.

Delay-Earliest-Due-Date

Este modelo não é mais do que uma extensão do modelo *Earliest-Due-Date-First*, em que a cada pacote a partir de um fluxo de tráfego é atribuído um tempo limite (*deadline*) e os pacotes são transmitidos por ordem crescente desse tempo limite. No novo modelo o servidor negoceia o contrato de serviço com a fonte em que este garante que se a fonte obedecer às especificações acordadas é garantido o atraso. Nesta situação, o servidor atribui um tempo limite ao pacote k, obtido da seguinte forma

$$ExD_{i,j}^{k} \leftarrow max\{a_{i,j}^{k} + d_{i,j}, ExD_{i,j}^{k} + X_{\min_j}\} \tag{4.10}$$

em que $d_{i,j}$ é o atraso local atribuído à conexão e $X_{\min_j}$ representa o tempo mínimo de chegada entre pacotes, tal que o pacote k seria servido se tivesse sido recebido de acordo com o contrato.

4.2.4 Disciplinas de serviço Non-Workconserving

Stop-and-Go

Esta disciplina de serviço utiliza uma estratégia de *framing*, isto é, divide o tempo em *frames* ou *slots* que não são mais do que períodos de tempo de comprimento fixo, por exemplo, T. Esta disciplina define uma *frame* de chegada e outra de partida para cada um dos *links*, em que as *frames* de chegada das ligações de entrada são mapeadas nas *frames* de saída de cada ligação através da introdução de um atraso constante, por exemplo, θ em que $0 \leq \theta < T$. A disciplina de serviço garante que os pacotes na mesma *frame* da mesma fonte se mantêm na mesma *frame* dentro da rede. Nesta aproximação o instante de tempo em que o pacote fica elegível é

$$e_{i,j}^{k} = a_{i,j}^{k} + Ahead_{i-1,j}^{k} + \theta \tag{4.11}$$

em que $a_{i,j}^{k}$ é o instante de tempo de chegada do pacote e $Ahead_{i-1,j}^{k}$ é o tempo que o pacote tem de esperar no comutador corrente antes de se tornar elegível.

Uma das características desta disciplina de serviço é ter um "controlador" de débito ligado ao escalonador e, por exemplo, se o controlador utilizado for (r, T) significa que não são transmitidos mais do que $r \times T$ *bits* durante uma *frame* de tamanho T. Desta forma é possível manter o padrão de tráfego desde a entrada da rede até ao destino.

Contudo, esta estratégia também apresenta alguns problemas, nomeadamente no limite do atraso e na distribuição da largura de banda, uma vez que não é possível alcançar simultaneamente um limite reduzido no atraso e uma granularidade "fina" da distribuição da largura de banda como é demonstrado em [122]. A resolução deste problema, segundo o mesmo autor, foi realizada com o recurso a uma versão mais generalizada do modelo através de *frames* com vários tamanhos; no entanto, o problema identificado mantém-se dentro das *frames* e em [101] é apresentada uma solução para a resolução do mesmo.

Hierarchical Round Robin

A disciplina de serviço HRR vem na linha da proposta discutida anteriormente porque usa uma estratégia de várias *frames* com vários níveis, em que uma *frame* de um nível pode ser atribuída quer para uma conexão ou para uma *frame* de nível inferior. Nesta aproximação o instante de tempo em que o pacote fica elegível é

$$e_{i,j}^{k} = \max(a_{i,j}^{k} + \tau, e_{i,j}^{k-q_{i,j}^{m}} + T_m) \tag{4.12}$$

em que $a_{i,j}^{k} + \tau$ é o instante de tempo de início da próxima *frame* e $q_{i,j}^{m}$ o número máximo de pacotes que podem ser servidos numa conexão dentro de cada *frame* de tamanho T_m.

Da mesma forma que a disciplina *Stop-and-Go*, HRR mantém dentro da rede o padrão de tráfego controlado devido à sua natureza N-WCing; no entanto, esta disciplina de serviço apresenta uma diferença fundamental em relação à anterior relativamente à forma como os pacotes viajam dentro da rede, isto é, enquanto no *Stop-and-Go* os pacotes viajam sempre na mesma *frame* até à saída da rede no HRR não viajam necessariamente na mesma *frame*. Esta disciplina apresenta os mesmos problemas da anterior devido à utilização da estratégia de *framing*.

Jitter-Earliest-Due-Date

Esta disciplina de serviço é uma extensão da disciplina *Delay-EDD*, mas com a capacidade de fornecer limites na variação do atraso. Basicamente, a disciplina opera da seguinte forma: após um pacote ter sido servido em cada servidor é atribuído uma etiqueta a um dos campos do cabeçalho com a diferença entre o seu tempo limite e o seu tempo final de serviço.

Assim que o pacote chega ao servidor a jusante ele é retido pelo regulador durante o tempo calculado no servidor a montante até que atinja o instante de tempo em que é considerado elegível para ser escalonado e assim sucessivamente até chegar ao destino. Nesta aproximação o instante de tempo em que o pacote fica elegível é obtido da seguinte forma

$$e_{i,j}^k = a_{i,j}^k + Ahead_{i-1,j}^k \tag{4.13}$$

em que $a_{i,j}^k$ é o instante de tempo de chegada do pacote e $Ahead_{i-1,j}^k$ é o tempo que o pacote tem de esperar no comutador corrente antes de se tornar elegível.

Earliest-Deadline-Due-Delay Jitter

Esta disciplina de serviço é baseada no esquema apresentado em [39] que usa conexões de rotas fixas, conhecidas por canais de tempo real ou simplesmente canais, cujas rotas são escolhidas na altura do estabelecimento do canal. Com o objectivo de fornecer serviços de tempo-real, as fontes são "obrigadas" a declarar, durante o estabelecimento do canal, as características do seu tráfego e quais os requisitos de desempenho que desejam. O esquema original é composto por três componentes: um procedimento de estabelecimento, um mecanismo de escalonamento e um mecanismo de controlo de débito.

Uma vez que a política de escalonamento desta solução não adicionou nenhuma alteração ao modelo original, então o modelo descrito será o da versão original. O esquema de escalonamento é realizado nos nós da rede e baseia-se numa filosofia de tempo limite. A cada pacote dentro do nó é atribuído um tempo limite, isto é, para atingir o objectivo do atraso o pacote deverá ser servido antes de atingido esse limite. Sendo $d_{i,j}$ o limite do atraso atribuído ao canal i no nó j, um pacote que circula no canal e chega a esse nó no instante de tempo t_0, terá o seu tempo limite igual a $t_0 + d_{i,j}$.

O mecanismo de controlo de débito é usado para repor o padrão original do tráfego por causa de uma possível distorção devido à flutuação da largura de banda dentro da rede. Nesta solução o servidor realiza uma tarefa adicional em relação ao modelo original: escreve no cabeçalho dos pacotes a diferença de tempo entre o instante em que o pacote é servido e o instante em que ele supostamente seria servido (tempo limite), informação que será necessária para o próximo regulador a jusante. Admitindo

que $J_{i,j}$ é o limite da variação do atraso no nó local e atribuído ao canal i do nó j, é usado para calcular o tempo elegível que é igual a $t_0 + d_{i,j} - J_{i,j}$. O pacote fica retido no regulador durante o tempo elegível calculado e só depois é transferido para o escalonador.

Rate-Control Static-Priority Queueing

O mecanismo *Rate-Control Static-Priority Queueing* (RCSP) é constituído por dois mecanismos: um controlador de débito e um escalonador. Este dois módulos são responsáveis por distribuir a largura de banda e os atrasos pelas diferentes conexões que atravessam os nós da rede. Esta combinação separa a distribuição da largura de banda e dos atrasos e simplifica o controlo da admissão de novas conexões e a implementação. A especificação do tráfego utilizada neste mecanismo é o (X_{min}, X_{ave}, I, P)[3] e é com base nesta especificação que o controlador de débito vai operar e dar garantias de QoS.

Este mecanismo oferece dois tipos de reguladores: (i) *Rate-Jitter* (RJ) que controla o *rate-jitter* através da reconstrução parcial do padrão de tráfego e (ii) *Delay-Jitter* (DJ) que controla o *delay-jitter* reconstruindo na totalidade o padrão de tráfego. Para o regulador RJ é definido um tempo elegível, ET_j^k do pacote de ordem k na conexão que atravessa o comutador j, com referência aos tempos elegíveis dos pacotes que chegam ao comutador na mesma conexão, da seguinte forma

$$ET_j^1 = AT_j^1 \tag{4.14}$$

$$ET_j^k = max(ET_j^{k-1} + X_{min} + \tau_j^k, AT_j^k),\ k > 1 \tag{4.15}$$

em que AT_j^k é o tempo de chegada do pacote k ao comutador j e τ_j^k é o número mínimo tal que o intervalo médio entre chegadas de pacotes em qualquer período I não excede X_{ave}. Para o regulador DJ, ET_j^k é definido com referência ao tempo elegível do mesmo pacote no comutador anterior:

$$ET_0^k = AT_0^k \tag{4.16}$$

[3] em que X_{min} representa o tempo mínimo de chegada entre dois pacotes, X_{ave} o tempo médio de chegada entre pacotes durante qualquer intervalo de tamanho I e P o tamanho máximo do pacote.

$$ET_j^k = ET_{j-1}^k + d_{j-1} + \pi_{j-1,j}, \quad j > 0 \tag{4.17}$$

em que o comutador 0 é a fonte que gera o tráfego, d_{j-1} o limite do atraso (ou tempo máximo de espera do pacote da mesma conexão no escalonador do comutador $j-1$) e $\pi_{j-1,j}$ é o atraso de propagação entre os comutadores $j-1$ e j.

O escalonador é composto por um número de filas prioritizadas de pacotes e uma única fila para pacotes sem requisitos de tempo real. Uma conexão é atribuída a um nível de prioridade no momento do estabelecimento da mesma. O escalonador serve os pacotes pela ordem FCFS das filas não vazias de prioridade mais elevada e só quando não existir nenhum pacote nas filas prioritizadas são servidos os pacotes da fila de tráfego sem requisitos especiais.

Rate-Controlled Static Priority

Esta disciplina de serviço tem como objectivo principal fornecer flexibilidade na distribuição da largura de banda e no controlo dos atrasos aliada à simplicidade de implementação. Neste tipo de disciplinas os mecanismos estão divididos em dois, um controlador de débito e um escalonador. No primeiro caso, o controlador de débito é constituído por um conjunto de controladores que controlam cada uma das conexões que atravessa o equipamento e é responsável por modelar o tráfego de entrada para que este não se desvie do padrão original.

O funcionamento desta disciplina é simples e processa-se da seguinte forma: assim que chega um pacote é calculado e atribuído ao pacote, pelo regulador, um tempo elegível que o vai obrigar a permanecer no regulador até que esse tempo elegível se esgote e esteja na cabeça da fila do escalonador para ser servido. Existem várias métodos de cálculo deste tempo, destacando-se, por exemplo, o regulador RJ com uma caracterização de tráfego dada por (X_{min}, X_{ave}, I) que assegura que o tráfego à sua saída satisfaz as condições estabelecidas pelo regulador, enquanto que o regulador DJ assegura que o tráfego à saída do regulador tem exctamente o mesmo padrão do tráfego que saiu do regulador a montante [121].

O escalonador selecciona o pacote que está à cabeça da fila com maior prioridade, em que estes níveis de prioridade correspondem aos vários níveis de atraso pelo que é possível tratar o tráfego de forma diferenciada.

4.2.5 Algumas considerações

Na apresentação das várias disciplinas de serviço pertencentes às duas classes (WCing e N-WCing) ficou claro que ambas apresentam vantagens e desvantagens; no entanto existem algumas características que devem ser melhor discutidas relativamente a cada uma das classes.

Para uma compreensão mais correcta dos problemas dos esquemas da classe WCing, é apresentado o exemplo utilizado em [75] que ilustra de forma clara as desvantagens descritas.

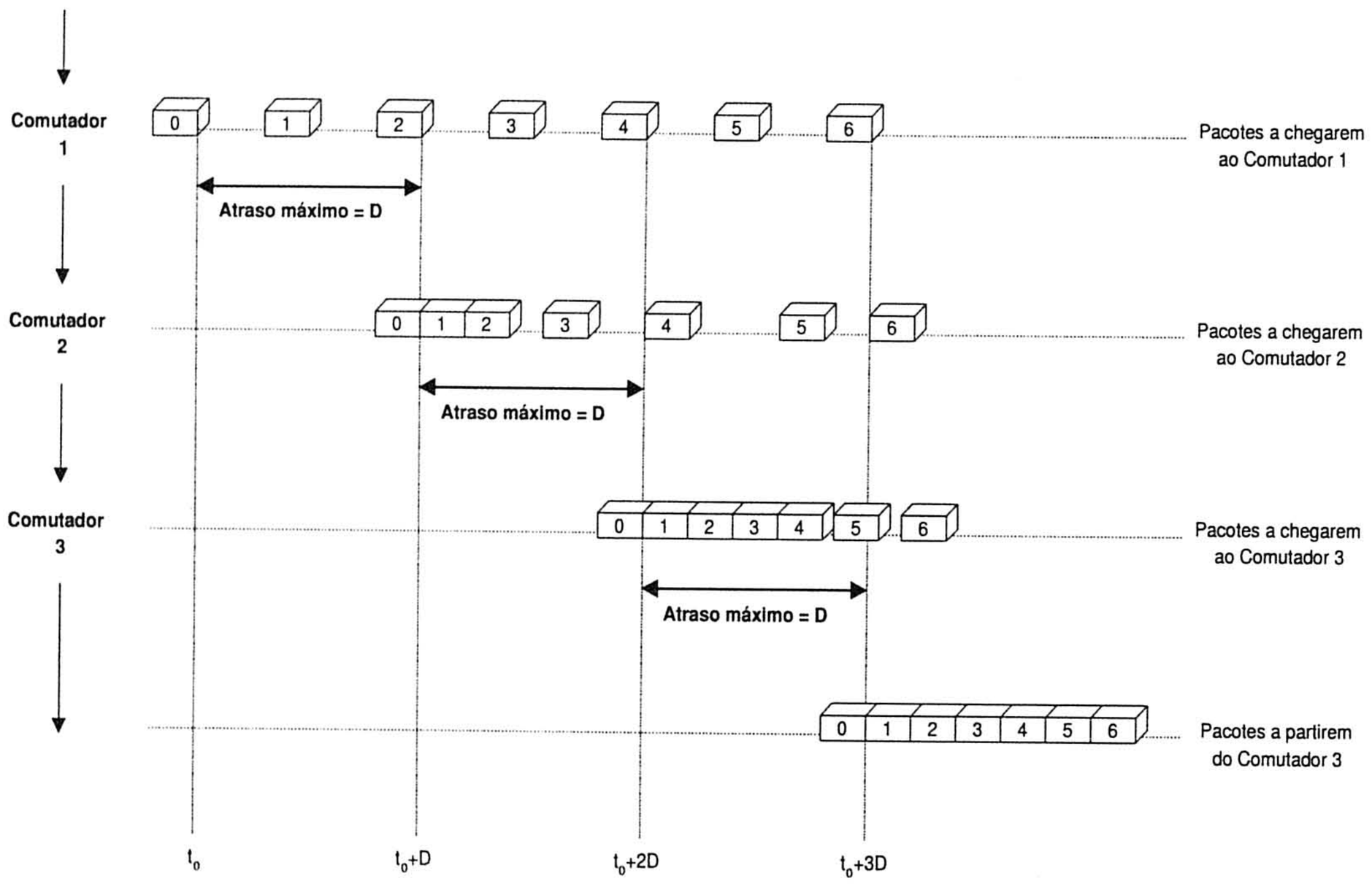

Figura 4.2: Chegada de pacotes a um escalonador *Work-conserving*.

Como pode ser observado na Figura 4.2 uma fonte transmite um conjunto de pacotes que atravessam três comutadores antes de chegarem ao destino. É assumido que o atraso máximo de cada pacote, em cada comutador, satisfaz um limite de atraso D. Nesta situação, os pacotes chegam ao primeiro comutador todos igualmente espaçados. O primeiro pacote vai sofrer o atraso máximo; é assumido que, em todos os comutadores, os pacotes serão enfileirados atrás do primeiro pacote e seguem-no até ao destino.

Na mesma figura o primeiro pacote, parte do primeiro comutador e chega ao segundo no instante de tempo $t_0 + D$. Nesse instante os pacotes 1 e 2 acumularam-se atrás do primeiro o que vai resultar num *burst* de 3 pacotes à chegada ao terceiro

comutador. Quando o primeiro pacote parte do 3^o comutador, no instante de tempo $t_0 + 3D$, o *burst* cresce e inclui todos os pacotes, de onde se pode concluir que por cada comutador atravessado o *burst* cresce.

Como conclusão, nas disciplinas de serviço da classe WCing o tráfego pode ser distorcido devido ao aumento do *burstiness* e consequentemente o aumento dos requisitos de espaço para os *buffers*. Como consequência ambas as situações agravam-se com o aumento do número de comutadores atravessados.

Em relação às disciplinas de serviço da classe N-CWing, como foi discutido na sua apresentação, é possível verificar que também existem vantagens e desvantagens na sua utilização. Analogamente ao exemplo anterior, é apresentado o exemplo (ver Figura 4.3) utilizado em [75] que ilustra o funcionamento genérico mostrando as suas virtudes e defeitos.

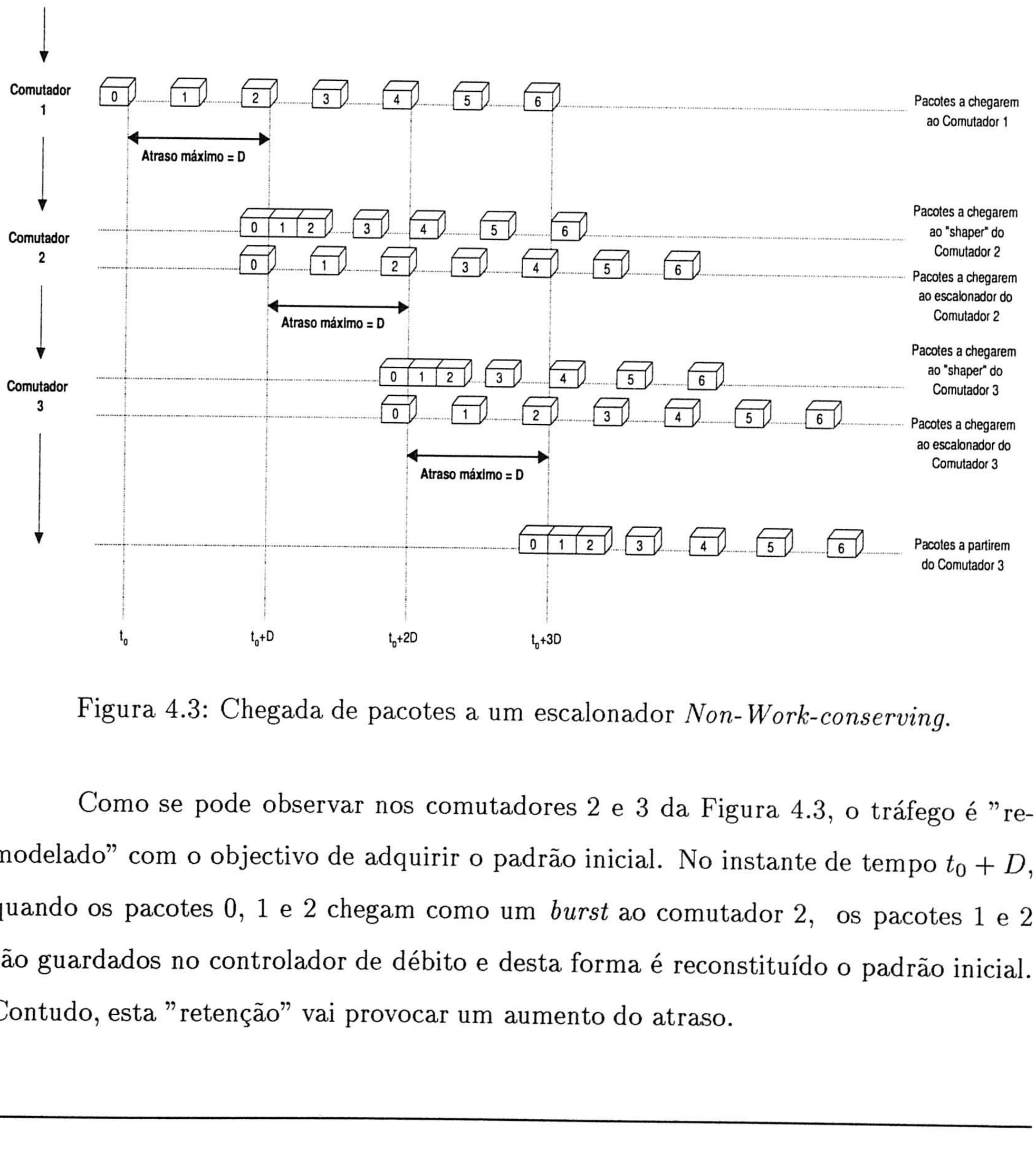

Figura 4.3: Chegada de pacotes a um escalonador *Non-Work-conserving*.

Como se pode observar nos comutadores 2 e 3 da Figura 4.3, o tráfego é "re-modelado" com o objectivo de adquirir o padrão inicial. No instante de tempo $t_0 + D$, quando os pacotes 0, 1 e 2 chegam como um *burst* ao comutador 2, os pacotes 1 e 2 são guardados no controlador de débito e desta forma é reconstituído o padrão inicial. Contudo, esta "retenção" vai provocar um aumento do atraso.

Com base na discussão à volta desta filosofia alguns autores apresentam em [70, 122, 25] um conjunto de vantagens e desvantagens na utilização desta classe de disciplinas de serviço. Resumidamente os argumentos contra a utilização de esquemas do tipo N-WCing são:

- A possibilidade de incrementar o atraso médio extremo-a-extremo; no entanto, se existiram aplicações que apenas desejem assegurar garantias quanto ao atraso máximo esta situação já não pode ser vista como um problema.

- Os controladores de débito "desperdiçam" largura de banda a não ser que o tráfego do tipo *best-effort* seja transmitido nos *slots* não utilizados.

- A necessidade de manter informação de estado por fluxo nos comutadores

- Uma fonte é sempre punida quando envia mais tráfego do que o especificado (por exemplo, para as aplicações que têm características de tempo real é difícil fazer uma caracterização correcta do padrão de tráfego antes da transmissão ter início).

Relativamente às vantagens (ou aos agrumentos a favor) para esta classe é possível indicar:

- A redução do montante total de espaço para os *buffers* necessário para um fluxo nos comutadores.

- Em contraste, os controladores de débito limitam o tamanho do *buffer*, mas em contrapartida aumentam o atraso médio.

- Para um serviço que garante limites no atraso, um escalonador com um controlador de débito pode satisfazer os mesmos limites de atraso que um escalonador do tipo WCing.

- Simplifica a análise no sentido de permitir derivar os limites de atraso em termos gerais, separa o controlo do atraso e a distribuição da largura de banda sem a necessidadede usar filas de prioridades ordenadas.

- Permite ter servidores heterogéneos com diferentes escalonadores e reguladores nos diferentes comutadores.

4.2.6 Considerações complementares

Adicionalmente aos testes efectuados por simulação sobre os modelos FIFO e WFQ (ver resultados no Anexo C) é possível provar que outro algoritmo bastante referenciado na literatura também apresenta problemas semelhantes relativamente aos objectivos que são propostos para este trabalho, nomeadamente, a distribuição equitativa da largura de banda e dos atrasos, assim como o controlo do *backlog*.

A solução apresentada em [112], conhecida por *Hierarchical Fair Service Curve* (H-FSC) tem como principal objectivo resolver o problema da obtenção simultânea de vários índices de desempenho, tais como garantir um débito mínimo e a distribuição da largura de banda remanescente segundo um critério de *fairness*, mas numa organização hierárquica, isto é, agrupar, por exemplo, os fluxos de vídeo, os de aúdio e os de dados em classes distintas e, num nível superior agrupar por instituição. O débito garantido para cada uma das instituições é subdividido pelas classes (atribuindo um débito mínimo garantido a cada classe) e após garantir o débito mínimo distribuir equitativamente a largura de banda remanescente. É provado nesta solução que é praticamente impossível conseguir na sua totalidade satisfazer os dois critérios em simultâneo para este tipo de organização.

A proposta H-FSC é apresentada para um tipo de organização que não é o caso que está a ser estudado nesta tese, mas apresenta problemas que são comuns como a distribuição da largura de banda remanescente e a garantia dos débitos mínimos negociados em conjunto com as questões relacionadas com a garantia de uma distribuição equitativa dos atrasos quando consideradas fontes do tipo *On-Off*.

Em [112] antes de ser apresentado o H-FSC foi utilizado uma solução baseada em *curvas de serviço* [4] que se designa por *Service Curve Earliest Deadline first* (SCED) [102], em que o tempo de transmissão de cada pacote é calculado usando uma curva de tempo de transmissão, $D_i(.)$, por sessão e os pacotes são transmitidos por ordem crescente dos tempos obtidos. Formalmente, o algoritmo SCED pode garantir todas as *curvas de serviço* se e só se $\sum_i S_i(t) \leq S(t)$ para qualquer $t \geq 0$, em que $S(t)$ é a quantidade de serviço que o servidor fornece no instante t, isto é, o somatório das *curvas de serviço* de todas as sessões deve ser menor ou igual à *curva de serviço* do servidor. Com base nesta ideia e na forma como o SCED realiza o escalonamento é possível

[4] O conceito de *curvas de serviço* foi discutido no Capítulo 3.

verificar que a distribuição não é equitativa e para mostrar que não é, o exemplo usado em [112][5] evidencia esse problema.

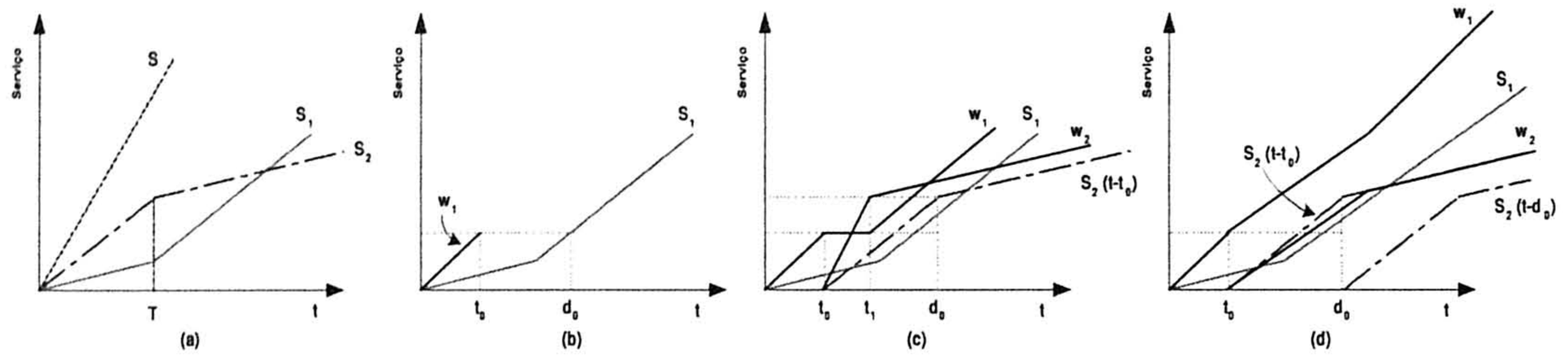

Figura 4.4: Exemplo de funcionamento da solução SCED e do seu melhoramento para H-FSC.

Embora o algoritmo SCED possa garantir todas as curvas de serviço simultaneamente não consegue ser equitativo na distribuição dos atrasos. Para provar esta afirmação será utilizado o exemplo da Figura 4.4. Como pode ser observado no gráfico (a) exsitem duas fontes que desejam transmitir com um débito mínimo garantido assegurado pelas *curvas de serviço* $S_1(.)$ e $S_2(.)$ (a *curva de serviço* do servidor é $S(.) \geq S_1(.) + S_2(.)$). Ambas podem ser garantidas usando o SCED, contudo, o servidor não garante em simultâneo os débitos de pico das duas fontes. A primeira fonte inicia a sua actividade e os valores de transmissão são calculados com a atribuição da *curva de serviço* à curva dos tempos de transmissão, isto é, $D_1(.) = S_1(.)$, (gráfico (b)) e transmite até ao instante de tempo t_0 (transmissão representada pela curva w_1), momento em que a segunda fonte inicia a sua actividade. Nesta situação é possível verificar que é assegurada a distribuição equitativa da largura de banda e o serviço atribuído é superior à *curva de serviço* ($w_1(.) > S_1(.)$).

Contudo, quando existe mais do que uma fonte para ser servida aparecem os mesmos problemas detectados com as disciplinas de serviço testadas por simulação (ver Anexo C), nomeadamente a falta de equidade na distribuição dos atrasos pelas fontes em competição. Esta situação acontece porque esta solução não tem em consideração a "história" passada (quantidade de serviço atribuída à fonte que estava a ser servida), logo a primeira fonte, como pode ser observado no gráfico (c) da Figura 4.4, vai ser prejudicada em relação à segunda fonte devido à forma como são calculados os tempos de transmissão dos pacotes (resultados confirmados mais tarde através do trabalho

[5] O exemplo apresentado é retirado integralmente do referido artigo.

apresentado em [90]).

A ideia de conseguir equidade na distribuição através da utilização do algoritmo SCED levou à alteração do cálculo dos tempos de transmissão dos pacotes. Como pode ser observado em certos momentos, o serviço atribuído pelo escalonador era inferior à *curva de serviço*, $S_2(t - t_0)$, (ver gráfico (d) da Figura 4.4), em que se verifica que a curva que representa o serviço atribuído à fonte 2 (w_2), sensivelmente entre t_0 e d_0, é inferior à *curva de serviço* ($S_2(t - t_0)$) que deveria ser garantida. Contudo, garante uma distribuição equitativa (como pode ser observado pelo paralelismo das curvas nesse período) devido ao facto de ter sido introduzido uma nova forma de cálculo dos tempos de transmissão dos pacotes, isto é, deixou-se de considerar que o início da *curva de serviço* era em $S_2(t - t_0)$ (instante de tempo em que a segunda fonte iniciou a sua transmissão ou ficou *backlogged*) e passou-se a considerar em $S_2(t - d_0)$; esta alteração leva o algoritmo a ter um comportamento idêntico ao WFQ.

Com base nesta dedução (extraída de [112]) e nos resultados obtidos (ver Anexo C) na secção C.3, em que se provou que o WFQ apresenta problemas na distribuição dos atrasos, é possível concluir que o algoritmo SCED com ou sem modificações na forma de calcular os tempos de transmissão dos pacotes também apresenta problemas semelhantes.

4.2.7 Conclusões

Na presente secção foram apresentados várias disciplinas de serviço, complementadas quer pelos resultados obtidos sobre seis cenários com diferentes características com duas disciplinas de serviço, FIFO e WFQ (ver Anexo C), quer pelos resultados da secção anterior.

Como observação global o comportamento obtido foi o esperado, quando se analisam os resultados com base no pressuposto que a garantia de serviço é conseguida dentro de um determinado período de tempo, mas não existe a preocupação de perceber os problemas e faltas de equidade que estão a ocorrer durante esse mesmo período. Durante a fase inicial de transmissão das fontes não se coloca qualquer problema do ponto de vista do controlo dos atraso e da ocupação dos *buffers*. A partir do momento em que as várias fontes estão a transmitir (os mecanismos testados, em especial a disciplina WFQ e SCED que são usadas como referência na literatura), perde-se a "história" do

que se passou e, por isso a distribuição dos atrasos não é equitativa. Poder-se-á afirmar que a partir de determinado momento, devido à falta de registo dos acontecimentos passados, o comportamento das disciplinas é semelhante e até pior no WFQ quando o número de fontes em competição aumenta.

A degradação de desempenho é acentuada quando existe um aumento do número de fontes em competição (de 2 para 4, por exemplo) além do aumento da decalagem do início da transmissão. É nesta situação que se verifica que as primeiras e as últimas fontes (relativamente ao tempo de início de actividade) são claramente beneficiadas em relação às restantes, porque não existe por parte das disciplinas a preocupação em compensar as outras da quantidade de serviço recebida pela primeira fonte enquanto esta era a única a transmitir e posteriormente a última que vai beneficiar da inexistência de células na fila de espera de entrada da primeira fonte. Este problema não se verifica apenas na distribuição dos atrasos, mas também na ocupação das filas de entrada. Através da ocupação das filas é evidenciada a injustiça entre as fontes que começam mais cedo e as que começam mais tarde; no entanto os atrasos ilustram outras injustiças, basta observar os últimos dois gráficos da Figura C.12 para se verificar e comprovar as afirmações.

4.3 Algoritmos de consolidação de células de gestão de recursos para conexões ponto-a-multiponto

Nas conexões ponto-a-multiponto existe já um número considerável de trabalhos apresentados devido ao facto do *standard* UNI 4.0 [24] contemplar um mecanismo para suportar tais conexões. Além deste facto muito trabalho tem sido desenvolvido relativamente ao controlo de fluxo e de congestionamento, com especial atenção à categoria de serviço ABR. Relativamente a esta categoria é possível suportar conexões ponto-a-multiponto através de sinalização e negociação de parâmetros de tráfego [77]; no entanto neste tipo de conexões aparecem os problemas referidos na secção 2.4.1 do Capítulo 2.

4.3.1 Controlo de congestionamento

Os mecanismos de controlo de fluxo em conexões ponto-a-multiponto não são mais do que uma extensão dos mecanismos ponto-a-ponto com a particularidade dos algoritmos da fonte e dos destinos não sofrerem qualquer tipo de alteração (ver Anexo D). Desta

forma pode ser utilizado um dos algoritmos de controlo de fluxo já existente (para ponto-a-ponto), mas apenas com ajustamentos para a nova realidade.

A afirmação realizada no parágrafo anterior é justificada com base nos esquemas propostos por alguns autores que apenas se concentram na resolução dos problemas relativos ao ponto-a-multiponto sendo utilizados esquemas já existentes para o cálculo do débito realizado pelos comutadores. Alguns dos autores que recorreram a esta metodologia foram, por exemplo, Sonia Fahmy [36] que utilizou o esquema ERICA [67], Wen Ren [95] que utilizou o esquema desenvolvido por Siu e Tzeng [107, 106], Hung-Shin Alex [19, 20] que utilizou um esquema para a distribuição do débito que é um misto do EPRCA [96] com o ERICA [67] e como será discutido no Capítulo 8 a proposta apresentada nesta tese recorreu ao algoritmo de distribuição da largura de banda desenvolvido pelo autor [81, 82].

4.3.2 Propostas

Nesta subsecção serão apresentadas e discutidas propostas de soluções para os pontos de agregação. Estas propostas serão apresentadas com base no funcionamento em termos gerais de uma comunicação ponto-a-multiponto da categoria de serviço ABR ilustrado na Figura 4.5.

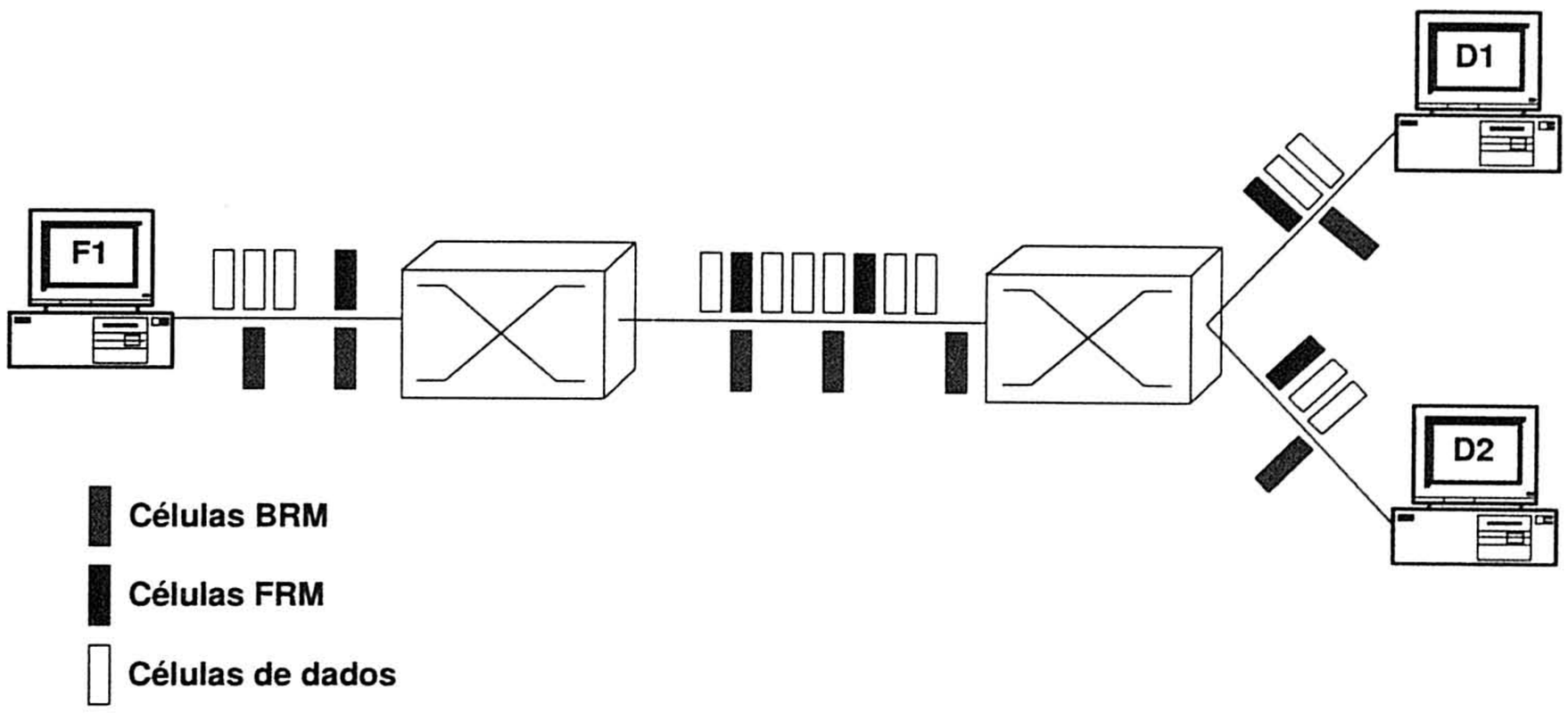

Figura 4.5: Configuração ponto-a-multiponto.

Como pode ser observado na Figura 4.5, no sentido fonte (F1) destinos (D1 e D2) circulam células de dados e de gestão de recursos designadas por FRM. Tanto as células de dados como a as células FRM ao chegarem ao segundo comutador são

copiadas para cada um dos ramos a jusante pertencentes à mesma conexão. As células FRM ao chegarem a cada um dos destinos são devolvidas para a fontes, mas agora com a designação BRM. A chegada destas células ao primeiro comutador (observando no sentido dos destinos para a fonte) introduzem os problemas discutidos no Capítulo 2 (*consolidation noise* e *feedback implosion*) e que são o motivo das propostas discutidas a seguir.

Lawrence Roberts

Lawrence Roberts [97] foi o primeiro autor a apresentar uma proposta para a resolução dos problemas das conexões ponto-a-multiponto. Neste mecanismo é proposto a introdução de uma variável designada por *Minimum Explicit Rate* (MER) no ponto de agregação (na Figura 4.5 é o comutador que tem duas ligações a jusante) que irá guardar o ER mínimo entre todos os ramos ligados a um determinado ponto de agregação, que servirá para informar a fonte do débito a que esta deve transmitir. Esta técnica é utilizada com vista a seguir de perto os requisitos das aplicações de dados onde não se deve perder informação.

Sempre que uma célula FRM é recebida, é replicada e transmitida para todos os ramos a jusante e uma célula BRM é gerada e enviada para montante com o valor de MER, sendo esta de seguida reinicializada. No entanto, o comutador gera uma célula BRM por cada FRM recebida, o que resulta numa desproporção entre o número de células FRM e BRM produzidas, para além de não reflectir o congestionamento a jusante e as células BRM não capturarem a informação de *feedback* de todos os ramos a jusante o que irá dar origem ao fenómeno *consolidation noise.*

Um melhoramento da solução anterior foi conseguido através da manutenção de uma *flag*, que obriga que seja gerada uma célula BRM apenas quando for recebida uma célula BRM de um ramo de jusante desde a última célula BRM enviada pelo ponto de agregação. Só assim é possível evitar o problema do aumento do tamanho dos *buffers*; no entanto é mantida a complexidade e o elevado processamento dentro dos comutadores.

Siu e Tzeng

Na proposta apresentada por Siu e Tzeng [106] é introduzido um registo no comutador designado por MER e um *bit* MCI para cada conexão. A variável MER é inicializada

com o primeiro valor transportado na primeira célula FRM que passa, enquanto que o *bit* MCI é inicializado a zero. Assim que o comutador recebe uma célula BRM o valor da variável MER guarda o débito mínimo entre todos os ramos de jusante. Da mesma forma o *bit* MCI é actualizado com o valor 1 se alguma célula BRM tem o campo CI atribuído a 1. No sentido oposto, quando uma célula FRM é recebida é realizada uma cópia dessa célula para todos os ramos a jusante do comutador. Contudo, uma célula BRM é retornada para montante com o campo MER e o *bit* MCI atribuído ao ER e ao campo CI da célula BRM. Após esta operação o comutador reinicializa a variável MER e o *bit* MCI.

O algoritmo apresentado contém alguns problemas, por exemplo, quando à variável que indica a recepção de uma célula BRM é atribuído o valor 1, independentemente de outras células BRM terem sido recebidas. Isto implica que a informação enviada para montante não seja rigorosa, isto é, não reflecte o estado do sistema. Por exemplo, durante uma iteração, o valor ER mínimo obtido entre o débito calculado no comutador e o transportado na célula BRM recebida nessa iteração vai ser enviado para montante numa célula BRM gerada assim que uma célula FRM for recebida. Desta forma é possível que a célula BRM do caminho mais congestionado esteja atrasada, pelo que só vai ser considerada numa próxima iteração e assim a informação fornecida à fonte por este esquema não é a mais correcta.

Wenge Ren e Siu

Weng Ren e Siu realizaram uma proposta [95] que resolve os problemas detectados no algoritmo de L. Roberts [97] e de Siu e Tzeng [106] através da manutenção de uma *flag* por cada porta de entrada para dar a indicação se por essa porta passou alguma célula FRM. Quando uma célula BRM chega a um comutador e após se realizarem os respectivos cálculos para a distribuição da largura de banda, é copiada para todas as portas que tenham a *flag* atribuída, sendo esta de seguida reinicializada.

Os mesmos autores propõem uma solução que reduz a complexidade do algoritmo através da passagem do retorno das células BRM dos pontos de agregação para os destinos. Este problema é resolvido através da atribuição de uma *flag* que indica a recepção de uma célula FRM. Quando uma célula BRM chega aos pontos de agregação é verificado se a *flag* está atribuída e se a resposta é positiva então a célula é tida em

consideração na obtenção do valor mínimo que seguirá na célula BRM em direcção à fonte [94]. Para além do problema descrito, esta solução continua a sofrer do problema de consolidação porque as células que são retornadas para montante não reflectem na totalidade a informação de jusante. Como na anterior, esta solução limita-se a enviar a célula BRM para montante sempre que recebe uma célula FRM, o que não resolve o problema descrito.

Sonia Fahmy, et. al

Sonia Fahmy, *et. al* em [36] argumentam que a modificação introduzida por Weng Ren, Siu e Suzuki diminui o *overhead* provocado pelas células BRM e é menos sensível aos níveis da árvore *multicast*, logo mais escalável relativamente ao atraso provocado pelas células BRM. Estes autores [36] propuseram três novos algoritmos (o segundo e o terceiro são evoluções do primeiro), em que a condição fundamental é a transmissão para montante de uma célula BRM recebida de jusante se o valor transportado pela célula BRM é muito menor do que o último valor do débito registado no comutador, ou se foram recebidas células BRM de todos os ramos de jusante ligados ao comutador. O algoritmo utiliza para o controlo de fluxo e de congestionamento o algoritmo ERICA [67] desenvolvido para as conexões ponto-a-ponto. Este algoritmo é "chamado" duas vezes, uma quando recebe uma célula BRM e a outra quando tem de transmitir uma célula BRM. Nesta solução é necessário conhecer previamente o número de ramos de saída de cada ponto de agregação para poder controlar o número de células BRM recebidas com o objectivo de reflectir para a fonte o débito correcto.

Os resultados apresentados em [36, 35] mostram o bom desenvolvimento das soluções propostas quando analisadas com base nos compromissos entre a complexidade e a minimização do problema de consolidação. Em [35] é mostrado que o algoritmo consegue manter o *max-min fairness*, controla o congestionamento e preserva a eficiência, mas cria problemas adicionais, tais como uma resposta lenta às transições para além de introduzir uma complexidade adicional. As conexões podem sofrer de uma resposta lenta do congestionamento mais distante provocando uma subutilização de um ramo por causa do congestionamento dos outros.

W. Melody Moh

O algoritmo desenvolvido por W. Melody Moh [79, 80] incide na resolução dos problemas detectados no algoritmo de Siu e Tzeng [106]. Para além da resolução desses problemas, introduziu ainda a possibilidade do algoritmo ser utilizado em redes ATM sem fios através da introdução de um relógio que permite evitar perdas e *deadlocks*. A grande diferença entre este esquema e o de Siu e Tzeng tem a ver com o tipo de condições utilizadas para desencadear a geração e envio de uma célula BRM para montante. Em primeiro lugar é enviada uma nova célula BRM quando o débito calculado no comutador é menor ou igual ao débito transportado na célula BRM recebida. Em segundo lugar é possível enviar para montante uma célula BRM, quando uma nova célula BRM foi recebida pelo comutador e a chegada de uma célula FRM faz com que o comutador calcule um novo débito que é menor ou igual ao débito corrente. Finalmente, é possível ainda enviar uma célula BRM para montante quando são recebidas todas as células BRM de todos os ramos de jusante a ele ligado, ou se pelo menos uma nova célula BRM foi recebida pelo comutador e um parâmetro de controlo de tempo Δ - tempo de *timeout* - terminou desde a última célula BRM ter sido enviada para montante.

Para implementar estas condições, o autor introduziu a *flag* $READY_p$ para cada caminho p da sessão i que atravessa um determinado comutador. Adicionalmente, foi necessário introduzir um relógio que determina quando a última célula BRM tenha sido enviada antes de Δ ter terminado, usado para evitar *deadlocks*.

Esta alternativa apresenta alguns problemas relativamente ao tratamento do congestinamento, isto é, não existe nenhuma indicação quanto ao tratamento deste tipo de problemas, para além de apresentar uma recuperação muito lenta nos momentos de transição devido à largura de banda disponível variar fortemente.

Hung-Shiun, Alex Chen e Klara Nahrstedt

As preocupações relativamente ao tempos lentos nas transições levou Hung-Shiun Alex Chen e Klara Nahrstetd a propor um novo mecanismo [19, 20]. O algoritmo introduz *timeouts* calculados de forma dinâmica.

Esta solução usa um algoritmo que ajusta o valor do *timeout* baseado na medida contínua do intervalo entre duas células BRM consecutivas. Cada ramo tem o seu

próprio limiar. O intervalo de espera de uma célula BRM de um ramo depende de a primeira célula BRM ser devolvida. O valor do *timeout* do ponto de agregação é definido como o máximo valor de todos os *timeouts* de todos os ramos a jusante. O intervalo vai ser determinado por dois parâmetros ABR: Trm e Nrm[6]. A ideia desta introdução é contemplar a possibilidade fornecida pelo UNI 4.0 [24], mais concretamente o *leaf initiated join* (LIJ), que não é mais do que permitir a entrada e saída de membros sem o conhecimento da raíz (ou fonte). Assim que um destino se junta à árvore *multicast*, o comutador mais próximo vai obter a informação *Fixed Round Trip Time* (FRTT) que é o somatório dos atrasos de propagação e fixos do comutador ao destino e voltar. O valor do *timeout* de um ramo é o tempo de espera mais longo para todos os destinos devolverem uma célula BRM. O comutador mede o RTT entre si e o comutador que se encontra a montante.

Esta solução não apresenta qualquer alternativa para tratar o problema do congestionamento, além de aumentar substancialmente a complexidade de implementação do mecanismo de consolidação e distribuição da largura de banda disponível.

4.3.3 Conclusões

Para as conexões ponto-a-multiponto já existe mais trabalho realizado do que para as outras conexões multiponto; no entanto existe ainda espaço para novas soluções porque, por um lado, ainda não existe nenhum *standard* aprovado pelo ATM Forum e, por outro lado, existe a necessidade de controlar os destinos que se encontrem muito distantes da fonte. Esta última questão levanta problemas quanto ao contolo do congestionamento, como por exemplo, deve-se ou não esperar em cada ponto de agregação por todas as células de *feedback* antes de ser enviada uma célula BRM para a fonte. Para além do problema referido é ainda necessário propor soluções que apresentem uma complexidade de implementação reduzida quando comparada, por exemplo com a complexidade do algoritmo proposto por Hung-Shiun, Alex Chen e Klara Nahrstedt [19, 20].

[6]Estes parâmetros encontram-se devidamente caracterizados na Tabela D.1 do Anexo D

Capítulo 5

Método para escalonamento e agregação de tráfego nos Dispositivos Periféricos

Neste capítulo é apresentado e discutido um novo método para escalonar fluxos individuais de tráfego que são de seguida agregados e partilham um canal comum, e que se caracterizam por serem modelados e policiados por um mecanismo do tipo *Token Bucket* (TB). Este método pode ser estendido para distribuir dinamicamente a largura de banda com base, não só na informação derivada do estado do TB, mas também na informação disponibilizada pela rede. A solução desenvolvida será validada com base na ferramenta matemática introduzida no Capítulo 3 e detalhada no Anexo A secção A.1, com vista a explicar as escolhas efectuadas relativamente às curvas de controlo, bem como à curva de serviço, para cada um dos fluxos a agregar. Após a justificação relativamente às escolhas é parametrizado o problema em estudo.

Na primeira secção é realizado um breve enquadramento do modelo, enquanto que na segunda secção são estabelecidos alguns pressupostos em que o método assenta e os objectivos a atingir. Nas duas secções seguintes é apresentada a arquitectura do ED e o modelo de tráfego utilizado. Na quinta secção são apresentados e discutidos os casos de referência. O método é apresentado e discutido na sexta secção. Na sétima secção são caracterizadas as *curvas de chegada*, de *serviço* e de *saída*, bem como a forma como estas curvas estão relacionadas com os parâmetros do modelo. Na secção seguinte

o problema é parametrizado e na última secção é apresentado um sumário do capítulo acompanhado das respectivas conclusões.

5.1 Introdução

A evolução nas tecnologias de comutação rápida de pacotes, impulsionadas pelo desenvolvimento das redes ATM e por avanços recentes das redes IP, mudaram consideravelmente o tradicional modelo de serviço *best-effort* usado em redes de comunicações de dados durante muitos anos. Este modelo está a ser completado com modelos que suportam a negociação e a provisão de QoS diferenciada, de forma previsível e mensurável, para fluxos de tráfego individuais e agregados.

Como explicado no segundo capítulo e sem demasiado formalismo, um TB é um contador não negativo que acumula *tokens* a um débito constante r, até eventualmente atingir a capacidade b. O TB pode ser usado para modelar/policiar pacotes ou fluxos, baseados em células ou pacotes de comprimento variável. Esta caracterização será feita com células, por ser mais simples relacionar directamente *tokens* com as células. Assim, uma célula é dita conforme (e consequentemente elegível para transmissão), quando pode reivindicar um *token* de um TB não vazio; nesse caso o contador é decrementado por cada nova célula que solicite um *token*. Quando o TB está vazio, as células ou são eliminadas ou retidas, até que sejam acumulados *tokens* suficientes. Quando o TB está cheio, os *tokens* entretanto gerados são ignorados. No modelo que será apresentado o TB (por fluxo, i) é parametrizado por um débito constante de renovação (geração) de *tokens* r_i e por um *bucket* de tamanho b_i, tal que

$$r_i = SCR_i \tag{5.1}$$

e

$$b_i = MBS_i \times \left(1 - \frac{SCR_i}{PCR_i}\right) \tag{5.2}$$

Os parâmetros utilizados na parametrização do TB são o SCR, o PCR e o MBS (notação do ATM Forum), mas apenas o valor do parâmetro SCR será utilizado na relação com a categoria de serviço ABR (devido à sua escolha para o transporte dos fluxos agregados), isto é, o somatório de todos os SCRs dos fluxos agregados será

mapeado no parâmetro débito mínimo garantido (MCR) do ABR. Como explicado no Capítulo 2, na categoria de serviço ABR, uma fonte pode negociar um MCR e solicitar um débito máximo que pode variar durante a ligação (PCR), enquanto que o débito corrente está limitado pelo débito permitido (ACR), indicado explicitamente pela rede e que se encontra entre MCR e PCR. Este débito é baseado geralmente numa distribuição da largura de banda disponível nas ligações congestionadas, de acordo com o critério *max-min fairness*. Uma das características interessantes desta categoria é o facto de as fontes receberem *feedback* da rede para adaptarem os seus débitos. Esta característica desempenhou um papel muito importante na escolha desta categoria de serviço, para ser usada na rede ATM para o tráfego em análise. Basicamente, os VCs ABR irão terminar nos equipamentos periféricos da rede de núcleo que funcionam, numa perspectiva extremo-a-extremo, como fontes e destinos virtuais. Como será observado no presente capítulo, estes VCs são do tipo ponto-a-ponto e, em capítulos posteriores, serão considerados conexões multiponto, nomeadamente ponto-a-multiponto, com o objectivo de rentabilizar o mais possível a utilização dos recursos da rede. É ainda assumido que a utilização da categoria de serviço ABR é realizada sempre com o parâmetro MCR maior do que zero para poder dar as garantias mínimas necessárias de QoS que as aplicações necessitam.

5.2 Pressupostos e objectivos

Uma das funções dos equipamentos periféricos (ED) é processar fluxos individuais e mapeá-los em fluxos agregados que partilham uma largura de banda comum (fixa ou variável, mas com um valor mínimo garantido, dependendo da categoria de serviço escolhida) e aos quais será dada uma garantia global de QoS. Com o propósito de resolver esta questão foi considerada uma classe particular de fluxos, em que os pacotes são modelados (e policiados) de acordo com os parâmetros do TB.

O modelo TB tem sido extensivamente estudado na literatura e apresenta-se como um mecanismo de modelação e policiamento, mas não é geralmente relacionado com a distribuição dos recursos e as políticas de escalonamento. O objectivo principal deste capítulo é explorar este relacionamento, usando os parâmetros do TB como uma base para as decisões na distribuição dos recursos e no método de escalonamento. O

método proposto é geral e, por isso, pode ser usado quando é escolhida uma categoria de serviço VBR ou ABR, para transportar fluxos agregados (uma vez que pressupõe uma largura de banda garantida com $MCR \geq \sum_i SCR_i$ quando a categoria de serviço escolhida é ABR e $SCR \geq \sum_i SCR_i$ quando é VBR).

Como se admite que a tecnologia usada na rede do núcleo é ATM, as células em conformidade com o TB são sujeitas ao escalonador que tem acesso ao estado do TB, com o objectivo de partilhar a largura de banda disponível. Em primeiro lugar, a largura de banda depende do serviço contratado e das exigências de desempenho; o ganho estatístico alcançado depende, entre outros factores, do número de fluxos multiplexados, do *burstiness* dos fluxos individuais e do grau de independência estatística entre fluxos. Os canais de débito variável, com algumas garantias mínimas, fornecem um grau mais elevado de flexibilidade e podem ser usados essencialmente para conseguir uma maior eficiência. Entretanto, sempre que o método é usado para reservar e distribuir a largura de banda é necessário o escalonamento de fluxos individuais que competem pela largura de banda partilhada. O escalonamento de fluxos individuais é realizado apenas no ED, antes de estes serem agregados, porque a partir desse momento deixam de ser vistos como tal. O método proposto tem como objectivo colocar estes aspectos - policiamento através do TB, distribuição instantânea da largura de banda e escalonamento de fluxos - num contexto comum.

5.3 Arquitectura

De acordo com a análise anterior, foi considerada uma rede com uma topologia com duas partes distintas, uma representando uma rede de acesso e a outra um *backbone* subdividido em rede de núcleo e rede de periferia, constituída por *Edge Devices* (EDs).

Esta topologia (ver Figura 5.1) é constituída por fontes e destinos que usam fluxos de uma classe de tráfego, tipicamente IP sobre diferentes tecnologias, quer na rede de acesso quer no núcleo. Com o propósito de agregar o tráfego IP da rede de acesso em VCs ATM da categoria de serviço ABR da rede de núcleo é necessário conceber uma arquitectura que permita atingir esse objectivo.

A Figura 5.2 ilustra a arquitectura proposta, e como pode ser observado a opção foi criar um módulo (bloco cinzento) que tivesse a função de escalonar e agregar tráfego

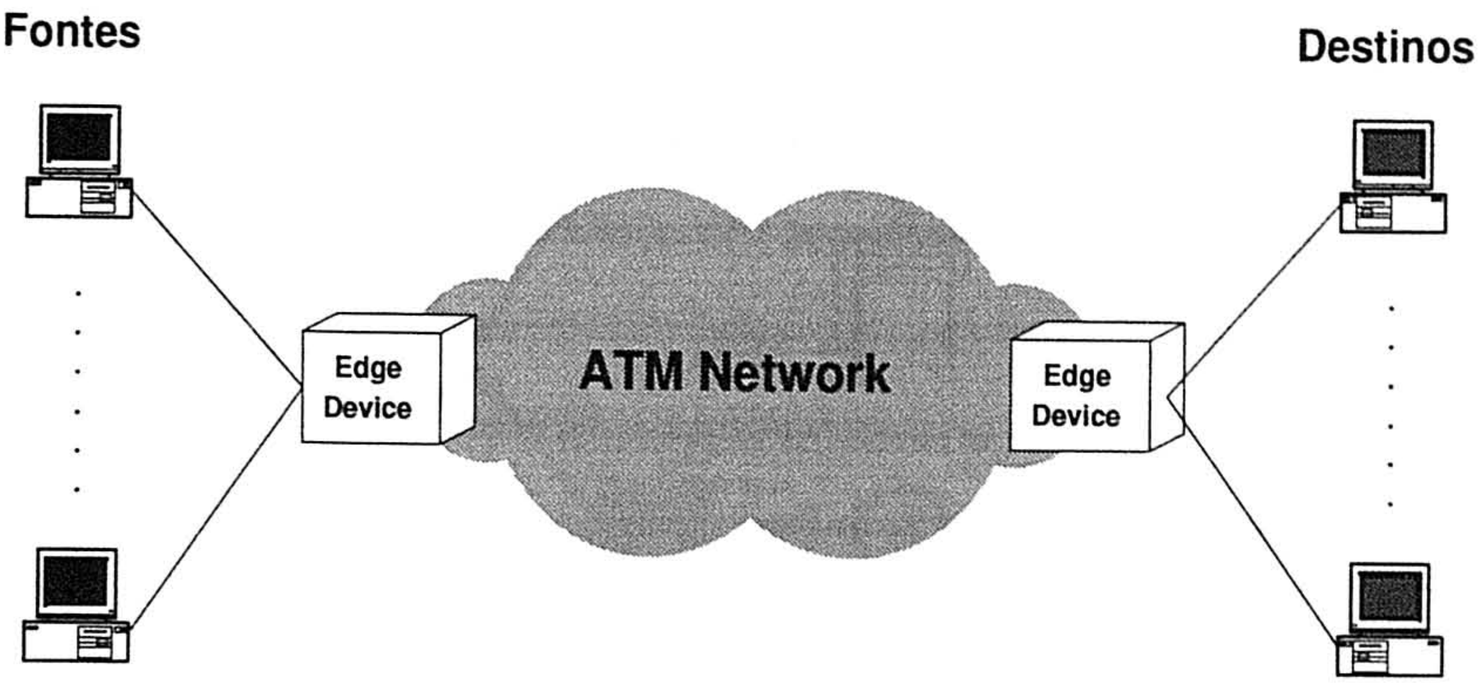

Figura 5.1: Configuração usada para explicar o método.

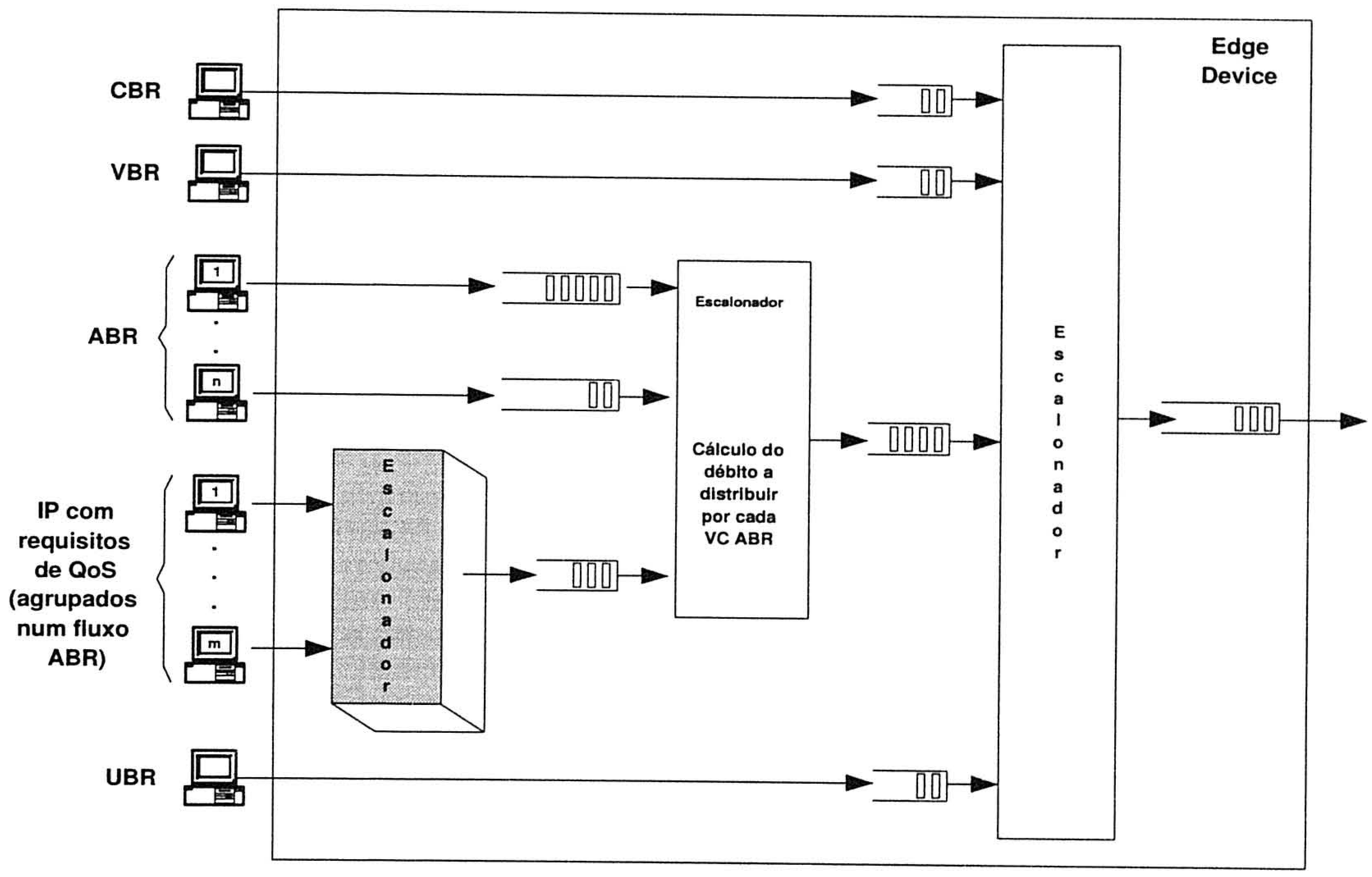

Figura 5.2: Arquitectura geral do *Edge Device* (1).

IP num único VC ABR. Este tráfego agregado vai competir no escalonador seguinte com VCs ABR, que podem transportar outro tipo de tráfego. O último escalonador serve para partilhar a largura de banda total, por todos os VCs de todas as categorias de serviço definidas para o ATM. No presente capítulo é apresentado o método de escalonamento dos fluxos IP num VC ABR, de acordo com os pressupostos e objectivos definidos inicialmente.

O método é aplicado ao módulo de escalonamento e agregação (bloco cinzento)

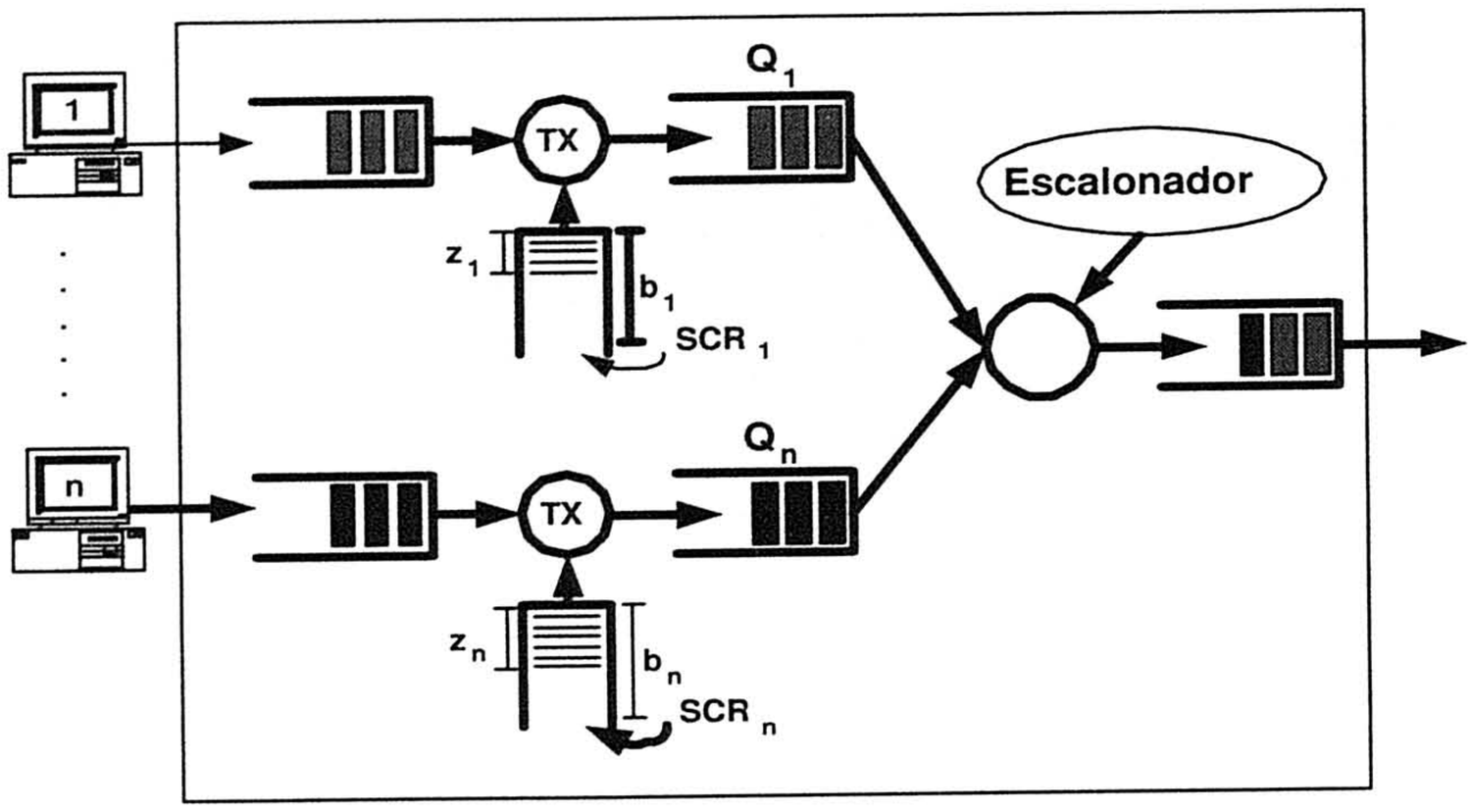

Figura 5.3: Arquitectura do módulo de agregação e escalonamento dos fluxos IP.

do ED que trata um número de fluxos controlados à entrada. Cada porta da entrada é constituída por um TB com um *buffer* a montante (não seria necessário se o fluxo tivesse sido modelado correctamente, isto é, o policiamento limita-se a confirmar a conformidade do tráfego) e um *buffer* a jusante para armazenar as células conformes que estão à espera de ser escalonadas. Todo o tráfego escalonado é agregado num *buffer* comum de saída (como ilustrado na Figura 5.3) que "alimenta" uma única conexão ATM partilhada por todos os fluxos (fila de saída do primeiro escalonador e de entrada do segundo).

Resumidamente, o escalonador selecciona a célula para ser transmitida dos *buffers* a jusante dos TBs de acordo com algumas regras que são baseadas nos estados individuais dos TBs, assim como no estado do sistema. Nas secções seguintes são apresentados casos de referência e um método de escalonamento (a estratégia proposta) que tem em consideração os aspectos referidos.

5.4 Modelo de tráfego

Para avaliação de soluções de distribuição de recursos ou de escalonamento é necessário recorrer a fontes de tráfego modelizadas de acordo com características e requisitos específicos das aplicações. Assim, é possível ter duas opções. A primeira (a mais complexa e que está fora dos objectivos deste trabalho) é o desenvolvimento de novos modelos (ou para melhor caracterizar o tráfego gerado por uma aplicação ou para modelizar o tráfego gerado por uma nova aplicação e que não representável por modelos existentes); por exemplo, o RealAudio [73] apresenta uma estrutura mais complexa do que os modelos

existentes, e por isso está ainda na fase de definição. A segunda é a utilização de um modelo existente, no caso deste trabalho é o TB, uma vez que este é representativo de uma ou mais classes de tráfego definidas em modelos de QoS aceites (ATM, IntServ e DiffServ).

A grande popularidade da Internet, a implantação da tecnologia de banda larga e o aparecimento de novos algoritmos de codificação de fontes audiovisuais, são factores que levam a que este tipo de tráfego (gerado pelas aplicações de aúdio e de vídeo), com requisitos específicos, continue a crescer, o que leva ao aparecimento de novas disciplinas de serviço. Estas disciplinas de serviço serão utilizadas em tráfego que tem de ser "conformado" (*shaped*) de acordo com o contrato que inclui os parâmetros de tráfego e de QoS (definidos no Capítulo 2). Uma vez que o contrato se baseia na negociação dos parâmetros PCR, SCR e MBS (ou o equivalente em IP) é obrigação da fonte manter o tráfego gerado dentro dos limites que serão policiados, porque deste ponto de vista para a rede é irrelevante a forma exacta da curva de tráfego, desde que esteja conforme. Seria, por isso, importante ter as curvas reais para dar mais "realismo" à simulação, mas do ponto de vista da validação dos algoritmos de escalonamento será suficiente ter curvas que possam servir de referência, desde que sejam "válidas" (conformes) e possam evidenciar situações críticas.

Da mesma forma que as soluções concebidas para escalonadores amplamente usados [30, 5, 111], utilizam o modelo de tráfego On/Off para testar as suas soluções relativamente aos atrasos e à distribuição da largura de banda, também neste trabalho, no desenvolvimento do escalonador, são utilizadas fontes como o mesmo modelo de tráfego. Estas fontes são parametrizadas de forma a constituirem a "envolvente" de tráfego conforme, em que os *bursts* são de duração máxima (fonte periódica), e para analisar algoritmos de escalonamento em diferentes situações são introduzidos diferentes graus de decalamento entre fontes.

5.5 Casos de referência

Antes de apresentar o modelo de escalonamento serão discutidos os objectivos a atingir, os cenários utilizados para obtenção/avaliação de resultados e alguns resultados de referência.

Serão utilizadas fontes do tipo On/Off periódicas e inicialmente idênticas, mas com vários graus de decalagem temporal. São considerados casos simples com duas, três e quatro fontes, relativamente aos quais é fácil, usando "diagramas temporais" determinar a solução alvo, de acordo com os seguintes objectivos: (i) "*fairness*" na distribuição dos atrasos sofridos pelas células e (ii) para fontes idênticas o atraso máximo deve ser idêntico.

Para atingir os objectivos definidos foram considerados vários cenários. De entre os diferentes cenários possíveis existem dois casos que não são contemplados nesta discussão, porque os resultados são evidentes e não apresentam dificuldades: fontes totalmente sincronizadas e fontes totalmente decaladas de forma que os períodos On nunca coincidem. Os cenários apresentados nesta secção baseiam-se na configuração da Figura 3.3, em que o número de fontes varia (entre 2 e 4) e com vários graus de correlação entre si.

O método apresentado servirá para escalonar células dos vários fluxos policiados/modelados por TBs. Um fluxo conforme é sujeito a limites conhecidos que podem ser facilmente ilustrados se forem consideradas fontes *On/Off*.

Considera-se que a fonte i transmite durante T_i^{On} um número de células igual a MBS_i, com débito PCR_i, e permanece inactiva durante um intervalo T_i^{Off}, tal que no início do próximo período possa novamente transmitir um MBS_i conforme, enquanto recebe um débito médio SCR_i durante todo o período (T_i^{Off} é o tempo mínimo para reabastecer o TB), sendo o período denotado por $T_i = T_i^{On} + T_i^{Off}$ (pode também ser obtido através da relação $\frac{MBS_i}{SCR_i}$). Os valores T_i^{On} e T_i^{Off} são dados por:

$$T_i^{On} = \frac{b_i}{PCR_i - SCR_i} \tag{5.3}$$

e

$$T_i^{Off} = \frac{b_i}{SCR_i} \tag{5.4}$$

Os valores dos parâmetros de configuração apresentados nas Tabelas 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, e em especial o número máximo de células (MBS), que para o primeiro cenário (Tabela 5.1) está fixado em 200 células, não foram escolhidos de forma a caracterizar um tráfego específico, mas apenas para testar a viabilidade e funcionalidade

do modelo que, naturalmente, não coloca restrições a esses valores. No entanto, sabe-se que, por exemplo, um pacote IP de comprimento máximo ocupa cerca de 200 células, logo não faria sentido ter um *burst* com a duração de apenas um pacote IP, mas do ponto de vista do ATM, este é um exemplo de um *burst* plausível. Contudo, se existisse a necessidade de caracterizar tráfego de vídeo ou aúdio teria de se usar valores de PCR e SCR mais reduzidos ou então aumentar o tamanho dos períodos de transmissão.

Análise prévia

De acordo com os objectivos definidos e com base num cenário constituído por duas fontes que iniciam a sua actividade decaladas de 5 ms e geram *bursts* com a duração de 20 ms é possível apresentar graficamente o comportamento ideal.

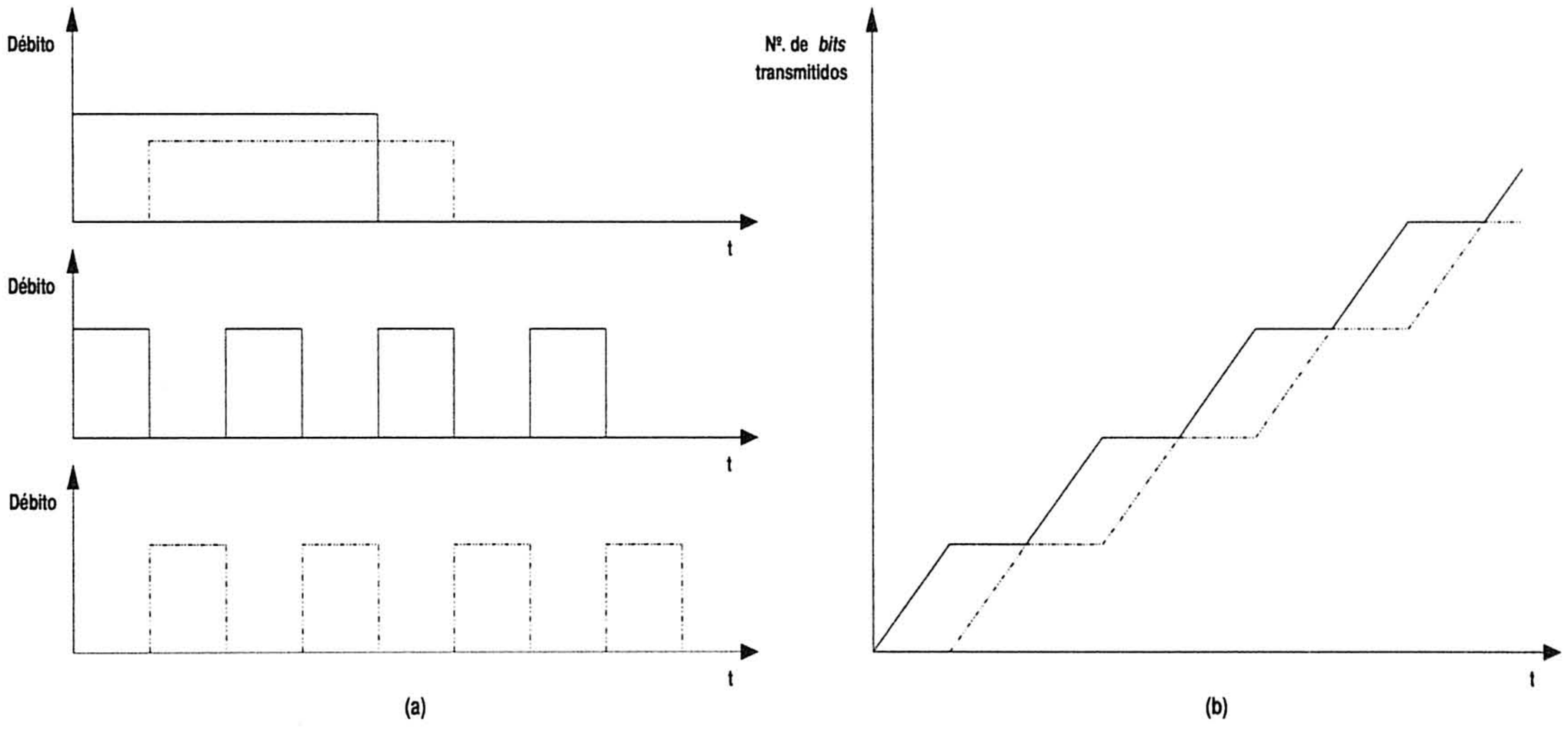

Figura 5.4: Resultados "óptimos" para uma configuração com duas fontes decaladas em um quarto do tempo.

Como pode ser observado no gráfico (a) da Figura 5.4, as duas fontes disputam a largura de banda, mas com o início dos seus *bursts* decalados. Nesta situação, a preocupação é apenas dividir a largura de banda pelas duas fontes, de modo a que exista equidade na distribuição dos atrasos e um controlo da ocupação dos *buffers*. A distribuição ideal é observada nos diagramas "temporais" da Figura 5.4 (a) e (b), relativamente a cada uma das fontes, através da atribuição de períodos de transmissão de comprimento fixo e intercalados entre cada uma delas.

Deve ainda ser referido que as fontes ilustradas na Figura 5.4 têm tratamento

absolutamente idêntico, isto é, o primeiro subconjunto de pacotes (cada subconjunto representa um quarto dos pacotes do *burst*) não sofre qualquer atraso, o segundo subconjunto sofre um atraso de $\frac{T_{On}}{4}$, o terceiro subconjunto sofre um atraso de $\frac{T_{On}}{2}$ e o quarto subconjunto sofre um atraso de $\frac{3T_{On}}{4}$ (representa o atraso máximo); no entanto se as células são servidas com um débito SCR então o atraso máximo seria $T_{Off} = T_{On}$.

De forma análoga ao cenário anterior, este cenário apresenta o mesmo número de fontes que geram *bursts* com a mesma duração, mas com o seu início de actividade decalado em 15 ms.

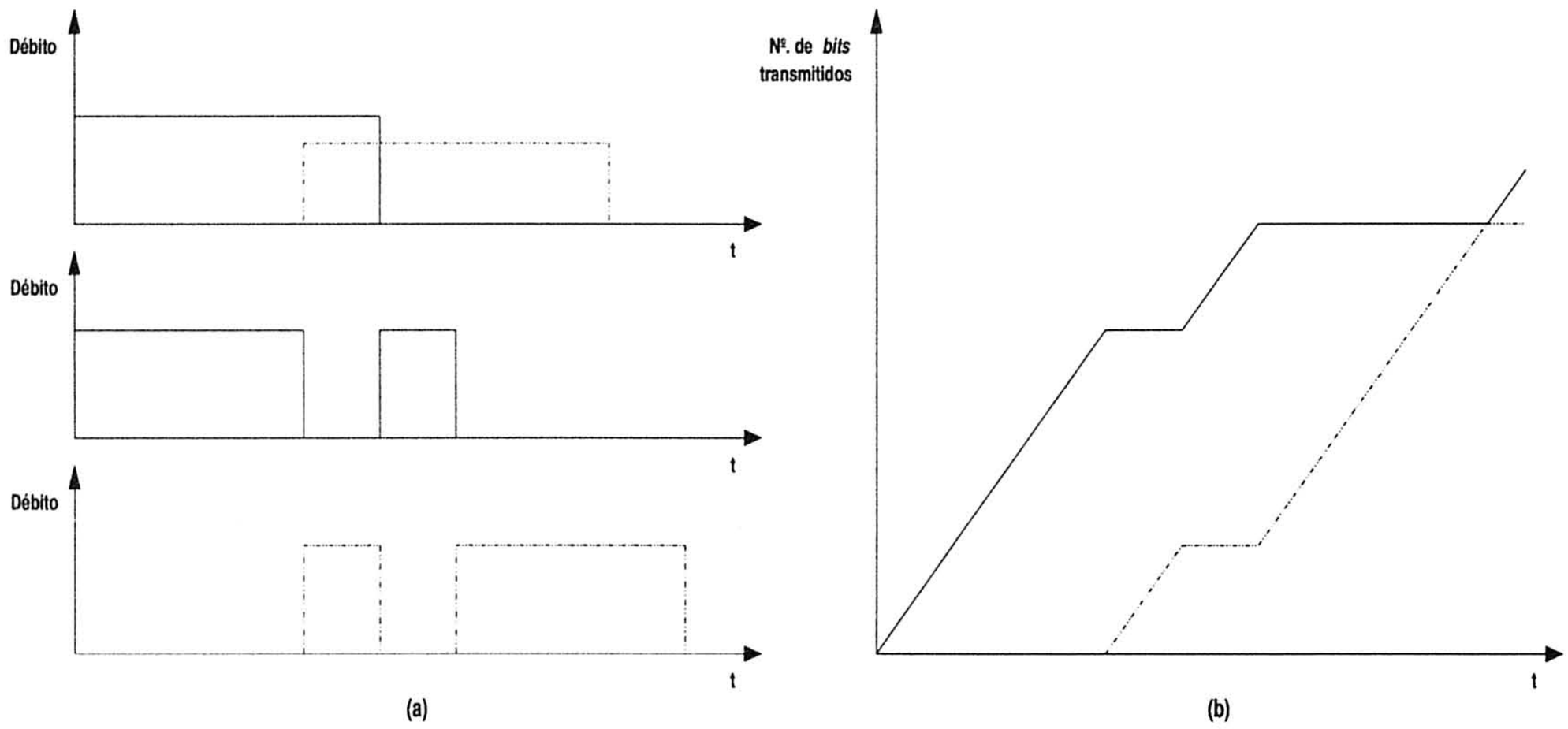

Figura 5.5: Resultados "óptimos" para uma configuração com duas fontes muito decaladas.

Nesta situação o atraso máximo (tomando como referência o segundo fluxo) é $\frac{T_{On}}{4}$, sendo este o objectivo a fixar para o primeiro fluxo, o que conduz à solução apresentada na Figura 5.5 (a) e (b).

Cenário 1

Este cenário é representado pela configuração da Figura 3.3 e pelos valores de configuração dos parâmetros apresentados na Tabela 5.1. Tem como objectivo estudar a reacção da disciplina de serviço à entrada em funcionamento de uma segunda fonte, quando uma já se encontra a transmitir; permitirá estudar, na fase mais básica, a distribuição da largura de banda entre as fontes em competição, a distribuição dos atrasos e a ocupação dos *buffers*. Uma das propriedades em estudo é observar em que momentos

é que a disciplina de serviço deixa de servir uma e passa a servir a outra fonte e verificar se os objectivos foram atingidos.

Tabela 5.1: Parâmetros de configuração para o Cenário 1

Fontes	*SCR* (Mbps)	*PCR* (Mbps)	*MBS* (Células)	*b* (Células)	T_{ON} (μs)	T_{OFF} (μs)	*Início de actividade* (μs)
F1	1	2	200	100	42400	42400	0
F2	1	2	200	100	42400	42400	5000

Cenário 2

O Cenário 2 é caracterizado pela mesma configuração do cenário anterior e pelos valores dos parâmetros fornecidos pela Tabela 5.2. Este cenário é testado com os mesmos objectivos do anterior, só que nesta situação, a segunda fonte inicia a sua actividade quando a primeira fonte transmitiu mais de metade do seu *burst* (cerca de $\frac{3}{4}$). Contrariamente ao primeiro caso, é impossível neste caso distribuir os atrasos de forma idêntica mas, deve garantir-se o mesmo atraso máximo.

Tabela 5.2: Parâmetros de configuração para o Cenário 2

Fontes	*SCR* (Mbps)	*PCR* (Mbps)	*MBS* (Células)	*b* (Células)	T_{ON} (μs)	T_{OFF} (μs)	*Início de actividade* (μs)
F1	1	2	200	100	42400	42400	0
F2	1	2	200	100	42400	42400	37400

Cenário 3

A configuração do Cenário 3 (ver Figura 3.3) é caracterizada por três fontes com a segunda e a terceira a começarem em simultâneo e decaladas da primeira. Os dados utilizados na configuração dos parâmetros são apresentados na Tabela 5.3. Este tipo de configuração permite estudar, para além das situações indicadas anteriormente, a forma como a largura de banda e os atrasos são distribuídos em relação às fontes que estão a transmitir em simultâneo e à fonte que já teve tempo de serviço atribuído.

Cenário 4

Neste cenário o objectivo (ver Figura 3.3 e Tabela 5.4) é verificar se o comportamento e os resultados obtidos são os mesmos do cenário anterior, uma vez que nesta situação, as

Tabela 5.3: Parâmetros de configuração para o Cenário 3

Fontes	SCR (Mbps)	PCR (Mbps)	MBS (Células)	b (Células)	T_{ON} (μs)	T_{OFF} (μs)	*Início de actividade* (μs)
F1	1	3	225	150	31800	63600	0
F2	1	3	225	150	31800	63600	10000
F3	1	3	225	150	31800	63600	10000

duas fontes que transmitem em simultâneo são substituídas por uma fonte que transmite com um débito igual à soma dos débitos das fontes que transmitem em simultâneo no caso anterior.

Tabela 5.4: Parâmetros de configuração para o Cenário 4

Fontes	SCR (Mbps)	PCR (Mbps)	MBS (Células)	b (Células)	T_{ON} (μs)	T_{OFF} (μs)	*Início de actividade* (μs)
F1	1	3	225	150	31800	63600	0
F2	2	6	450	300	31800	63600	10000

Cenário 5

O Cenário 5 considera a configuração da Figura 3.3, e os valores de configuração da Tabela 5.5 e tem como objectivo estudar o comportamento das disciplinas de serviço relativamente à escalabilidade, isto é, qual a reacção do método ao aumento do número de fontes. Neste cenário os tempos de intervalo entre as fontes é de 5 ms e vão servir para realizar um estudo similar ao do Cenário 1 com a complementaridade da escalabilidade.

Tabela 5.5: Parâmetros de configuração para o Cenário 5

Fontes	SCR (Mbps)	PCR (Mbps)	MBS (Células)	b (Células)	T_{ON} (μs)	T_{OFF} (μs)	*Início de actividade* (μs)
F1	1	4	200	150	21200	63600	0
F2	1	4	200	150	21200	63600	5000
F3	1	4	200	150	21200	63600	10000
F4	1	4	200	150	21200	63600	15000

Cenário 6

Com o mesmo propósito do cenário anterior, inclusivé com a mesma topologia (ver Figura 3.3) o Cenário 6 tem os mesmos objectivos, mas com intervalos de tempo maiores entre as fontes (ver Tabela 5.6), para perceber e analisar a reacção das disciplinas de serviço a esta situação.

Tabela 5.6: Parâmetros de configuração para o Cenário 6

Fontes	SCR (Mbps)	PCR (Mbps)	MBS (Células)	b (Células)	T_{ON} (μs)	T_{OFF} (μs)	*Início de actividade* (μs)
F1	1	4	200	150	21200	63600	0
F2	1	4	200	150	21200	63600	10000
F3	1	4	200	150	21200	63600	20000
F4	1	4	200	150	21200	63600	30000

Algumas considerações

Com base nos cenários definidos e nos objectivos estabelecidos foi realizado um conjunto de simulações para avaliar algumas disciplinas de serviço amplamente usadas, nomeadamente FIFO e WFQ (os resultados e a respectiva discussão podem ser observados no Anexo C). Destes resultados é possível concluir que a disciplina FIFO apresenta resultados que não são aceitáveis, quando analisados à luz dos objectivos definidos anteriormente; no entanto servem como referência para avaliar o WFQ. Relativamente aos resultados obtidos com o WFQ verifica-se que esta disciplina de serviço está longe de atingir os objectivos definidos.

5.6 O Modelo

Se a capacidade do canal de saída for representada por C e $R^{out} = C$ (admite-se inicialmente C constante) e assumindo que $C \geq \sum_i SCR_i$ então está garantido que o débito médio pode ser assegurado a todas as fontes (o TB garante por outro lado que o débito médio na entrada não é excedido). Esta condição tem um papel de estabilidade e garante que o *buffer* fica vazio, no pior dos casos, no fim do período T_i (se o débito de serviço atribuído a cada fonte for SCR_i).

Inicialmente é analisado o comportamento de uma única fonte On/Off, independentemente das outras, com um débito de saída de R_i^{out} e limitado por $SCR_i \leq R_i^{out} \leq PCR_i$. Assumindo que S_i^{On} é o tempo em cada período em que o *buffer* e a ligação são utilizados pela fonte i, enquanto que S_i^{Off} é o tempo em que a fonte i não está a utilizar qualquer recurso, então $S_i^{On} + S_i^{Off} = T_i$ e S_i^{On} excede T_i^{On} pelo tempo que o *buffer* demora a ser drenado da sua ocupação máxima (O_i) até ficar vazio. Sendo b_i a ocupação máxima possível (com $O_i \leq b_i$), R_i^{out} o débito atribuído pelo escalonador, o tempo de serviço atribuído à fonte i é

$$S_i^{On} = T_i^{On} + \frac{O_i}{R_i^{out}} \tag{5.5}$$

enquanto que o tempo de repouso é

$$S_i^{Off} = T_i^{Off} - \frac{O_i}{R_i^{out}}. \tag{5.6}$$

Consequentemente a fracção de tempo em que o sistema está ocupado com a fonte i é

$$s_i = \frac{S_i^{On}}{S_i^{On} + S_i^{Off}} \tag{5.7}$$

mas, como $T_i = S_i^{On} + S_i^{Off}$ a fonte i ocupa o sistema durante

$$s_i = \frac{S_i^{On}}{T_i} \tag{5.8}$$

e, da mesma forma, a fracção de tempo em que o sistema não "conta" com a fonte i é

$$s_i^* = \frac{S_i^{Off}}{T_i}, \tag{5.9}$$

logo a ocupação máxima do *buffer* é obtido da seguinte forma

$$O_i = \frac{b_i(PCR_i - R_i^{out})}{PCR_i - SCR_i} \tag{5.10}$$

(notar que O_i decresce linearmente com o aumento de R_i^{out}).

A partir do momento que se estabeleceram relações básicas de funcionamento do modelo para uma fonte é necessário definir as novas variáveis e condições que vão ser parte do seu funcionamento. Considera-se, em primeiro lugar, o caso em que ao fluxo é atribuído uma largura de banda constante e igual a SCR_i. Designa-se por Z_i o número de *tokens* disponíveis no TB i, por Q_i o número de células enfileiradas no *buffer* a jusante do TB i e como condição inicial $Z_i = b_i$ (*bucket* cheio). No instante de tempo $t = 0$, $Z_i = b_i$ e $Q_i = 0$, e quando se inicia o processo, o valor de Z_i diminui enquanto que Q_i aumenta e em $t = T_i^{On}$, $Z_i = 0$ e $Q_i = b_i$. Nesta situação a fila é drenada com débito SCR_i; no instante $t = T_i^{On} + T_i^{Off}$ as condições iniciais restabelecem-se. Em qualquer instante de tempo a condição $Z_i + Q_i = b_i$ é verificada. Contudo, se a largura de banda do canal é $h \times SCR_i$ $(h > 1)$, Q_i aumenta com um ritmo menor e a fila é drenada mais

rapidamente porque se verifica a condição $z_i + Q_i < b_i$. Seja $X_i = b_i - (Z_i + Q_i)$; neste caso $X_i > 0$ (enquanto $X_i = 0$, no primeiro caso) significa que o fluxo beneficiou de uma largura de banda superior a SCR_i.

É possível dar uma interpretação dos parâmetros acima utilizados, mas antes dessa interpretação é necessário fazer a distinção entre *tokens* para policiamento e *tokens* para transmissão. No primeiro caso os *tokens* são atribuídos pelo TB às células que vão chegando, desde que estas se encontrem conformes, e estas são passadas para o *buffer* a jusante, enquanto que no segundo caso é o escalonador que atribui *tokens* que serão utilizados para servir as células dos *buffers* a jusante dos TBs. Os parâmetros utilizados no método são:

- Z representa o número de *tokens* disponíveis no TB; um Z grande significa que para tráfego conforme pode vir a ocorrer um *burst* longo, enquanto que um Z pequeno significa que apenas se podem esperar *bursts* pequenos (tendo o sistema evoluído para este estado de diferentes maneiras possíveis, por exemplo na sequência de um *burst* longo ou de forma regular);

- $b - Z = Q + X$ representa o número de *tokens* usados e portanto células que podem ser escalonadas; estas requerem *tokens* para transmissão (*slots* de escalonamento);

- X representa o número de *tokens* de transmissão que foram atribuídos em excesso (ou em avanço) relativamente a uma atribuição baseada no SCR (média);

- Q representa o número de *tokens* de transmissão ainda não atribuídos (células enfileiradas que passaram a conformidade, mas que ainda não foram escalonadas, isto é, aguardam *tokens* de transmissão).

Na Figura 5.6 estão representados os parâmetros definidos anteriormente (Z_i, X_i, Q_i), a fonte *On-Off* é representada por OBD, enquanto que OBC representa o limite superior para um fluxo conforme a um TB e OD representa um serviço com débito constante SCR_i. Estão ainda representados R_i^{in} e R_i^{out} que correspondem a situações intermédias. $R_i^{in} \times t$ e $R_i^{out} \times t$ representam a acumulação de *bits* recebidos/transmitidos no caso de débito de entrada e de saída constantes; em geral R_i^{in} e R_i^{out} variam instantaneamente pelo que os *bits* acumulados são

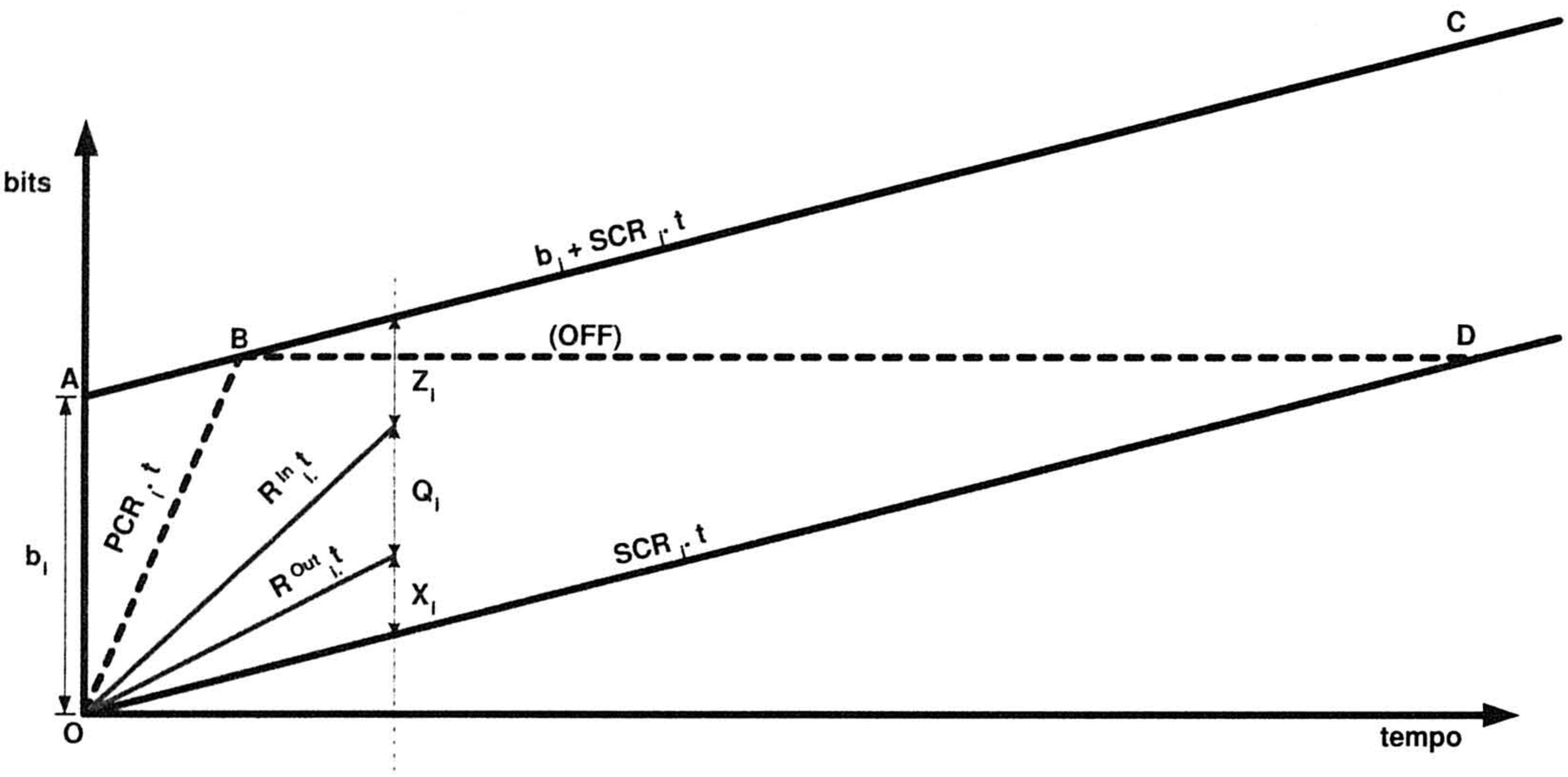

Figura 5.6: Representação gráfica dos parâmetros do método.

$$A_i^{in}(t) = \int R_i^{in} dt \tag{5.11}$$

e

$$B_i^{out}(t) = \int R_i^{out} dt, \tag{5.12}$$

em que $A_i^{in}(t)$ e $B_i^{out}(t)$ são funções não decrescentes e $A_i^{in}(t) \geq B_i^{out}(t)$; no entanto em alguns momentos é possível ocorrer $R_i^{out} > R_i^{in}$, isto é, $\frac{dB_i^{out}(t)}{dt} > \frac{dA_i^{in}(t)}{dt}$.

Para uma melhor compreensão dos parâmetros envolvidos e da sua relação, a Figura 5.7 é útil na medida em que mostra como os valores evoluem ao longo de uma simulação para uma fonte. Esta figura mostra a relação existente entre o período de transmissão do *burst*, designado por T_i^{On} e o tempo de serviço concedido, designado por S_i^{On}. Verifica-se que, por um lado, se R_i^{out} tende para PCR_i a ocupação máxima tende a baixar, tal como o valor de S_i^{On}; por outro lado, se R_i^{out} tender para SCR_i a ocupação tende para o seu valor máximo (b_i) e S_i^{On} para $T_i^{On} + T_i^{Off}$.

Quanto ao comportamento do valor de Z_i este aumenta e diminui de forma linear com o evoluir do tempo de actividade, isto é, vai diminuindo até zero enquanto a fonte está a transmitir o seu *burst* e aumenta até que seja atingido o valor máximo de ocupação ($Z_i = b_i$), durante o tempo em que a fonte estiver em repouso, para que no final do período T_i se restabeleçam as condições que permitam suportar um novo *burst* máximo.

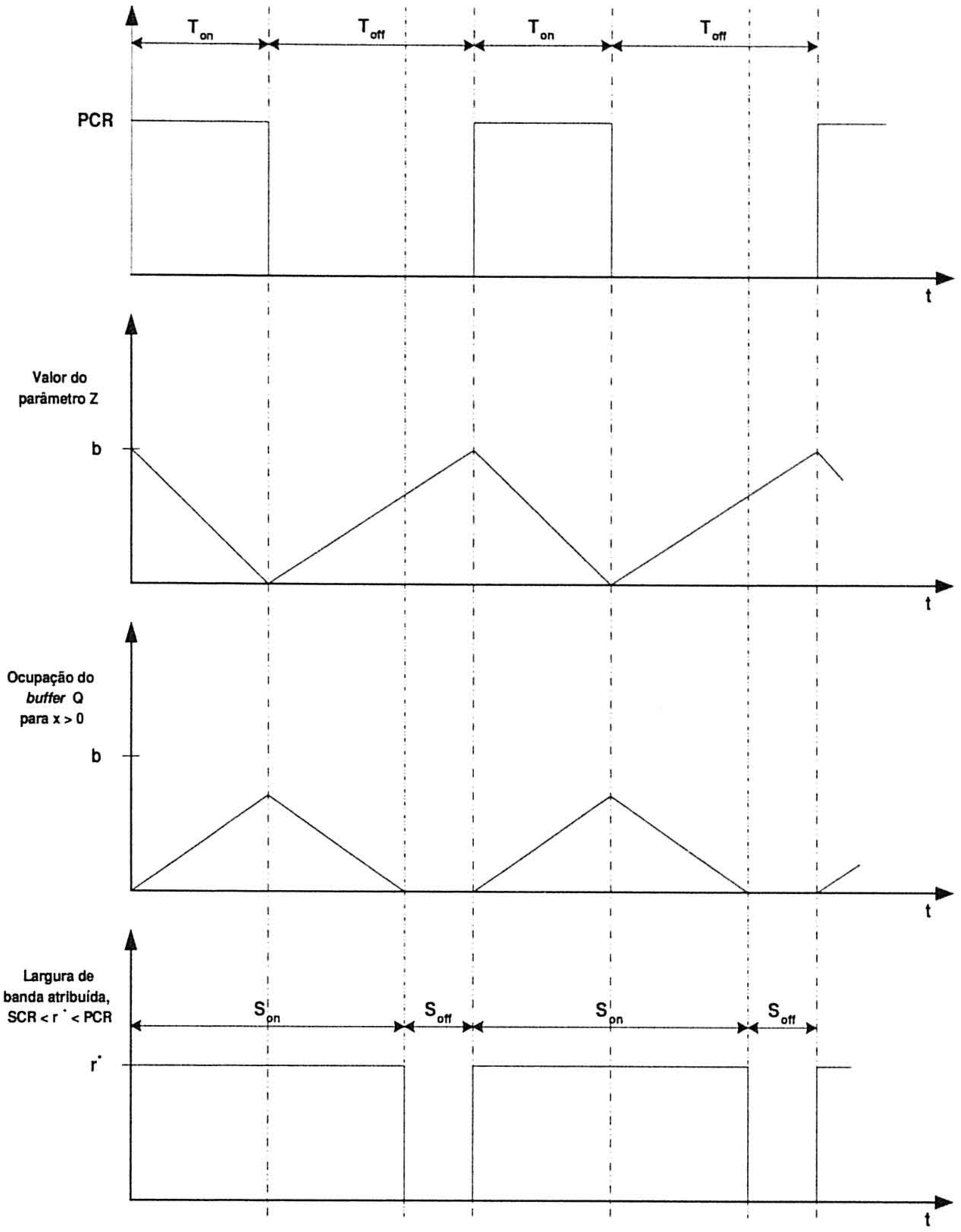

Figura 5.7: Relação entre os vários parâmetros utilizados no modelo.

Quando são multiplexados vários fluxos e, assumindo que eles são estatisticamente independentes, a largura de banda pode ser distribuída entre os fluxos através de uma política de escalonamento adequada. Como exemplo, é possível considerar um número de fontes *On-Off* em que (Z_i, X_i, Q_i) representam o estado de cada fluxo, enquanto (Z, X, Q) representa o estado global. Se os períodos *On* das fontes não se sobrepuserem (situação ideal), Z é constante. Esta situação significa que com uma largura de banda constante ($SCR = \sum_i SCR_i$) é possível escalonar os fluxos sem enfileiramento; neste caso, um canal CBR seria o melhor para conseguir o desempenho ideal. Pelo contrário, se os fluxos estivessem fortemente correlacionados (no limite períodos de actividade coincidentes), Z teria uma excursão máxima (entre 0 e $b = \sum_i b_i$); neste

caso uma largura de banda fixa distribuída equitativamente produziria um desempenho similar ao de um único fluxo com atribuição constante SCR_i. Esta situação sugere a exploração (se possível) de um canal VBR, com uma atribuição o mais elevada possível durante os períodos *On* e uma distribuição menor durante os períodos *Off*. Contudo, esta situação pode-se aplicar da mesma forma em conexões ABR desde que se assegure um valor não nulo do parâmetro MCR (e que deve ser pelo menos igual a $\sum_i SCR_i$).

A primeira conclusão a extrair desta discussão é que a excursão pico-a-pico de Z fornece uma informação valiosa relativamente a uma possível estratégia global de distribuição da largura de banda (isto é, do ponto de vista da partilha de largura de banda com outras classes do tráfego), mas o problema do escalonamento ainda se mantém. Ao contrário dos algoritmos de escalonamento descritos no Capítulo 4, este modelo propõe usar (em parte) o estado do TB como critério para escalonamento. Em primeiro lugar, o policiamento pelo TB assegura que os fluxos estão conformes antes de serem escalonados e, em segundo lugar, espera-se que os recursos sejam reservados (duma forma determinística ou estatística), de acordo com os parâmetros de tráfego declarados e as exigências de desempenho; é mesmo desejável que possam ser ajustados às exigências instantâneas, como sugerido. O objectivo é ter uma partilha equitativa da largura de banda disponível (qualquer que seja o seu valor) entre os fluxos, de acordo com seu estado real (tamanho da fila, quanta largura de banda acima da média foi usada, quanto tempo está "previsto" o *burst* durar, etc.), e não apenas quanto às exigências do débito médio ou de pico negociadas. Naturalmente a informação do estado do TB pode ser usada como uma aproximação a diferentes critérios e será explorada mais adiante.

5.7 Caracterização das *curvas de entrada*, de *serviço* e de *saída* do modelo

Para a caracterização das diversas curvas utilizadas no modelo proposto recorreu-se à ferramenta matemática desenvolvida por Le Boudec [12] que tem como base teórica o conceito de *Curvas de Serviço* introduzidas por R. Cruz [25, 26], como introduzido no Capítulo 3. A justificação detalhada relativamente às escolhas das *curvas de chegada*, de *serviço* e de *saída*, bem como a apresentação das definições e teoremas, será desenvolvida no Anexo A, secção A.2.

No modelo desenvolvido assumiu-se que o trafégo de entrada seria limitado superiormente quer pelo PCR instantâneo, quer pela curva de referência dos TBs, $\gamma_{r,b}$, e inferiormente pela *curva de serviço* λ_r.

Antes de continuar a caracterização é necessário definir que valores são atribuídos a R e a r para evitar qualquer confusão na sua interpretação, isto é, quando as funções de referência, por exemplo, λ_R, utilizam R é assumido que $R = PCR$ e quando utilizam r é assumido que $r = SCR$; o mesmo se verifica para qualquer uma das funções de referência utilizadas nesta tese.

5.7.1 *Curva de chegada*

Com base nesta ferramenta matemática é possível assumir que um fluxo de entrada é limitado/controlado por um TB, utilizando uma *curva de chegada* da forma $\alpha = \gamma_{r,b}$. Este tipo de curva permite a uma fonte enviar b *bits* num *burst*, mas não mais do que r *bit/s* ao longo do período T_i. Dado que foi assumida a utilização da função de referência $\gamma_{r,b}$, é necessário verificar, de acordo com a Definição 8, se é uma "boa" função, situação que se veio a verificar.

Aplicando a ferramenta matemática a este modelo é possível, através da convolução entre as funções λ_R e $\gamma_{r,b}$ (com $r = SCR_i$), verificar que

$$A^{in} \leq A^{in} \otimes \alpha \Longleftrightarrow \lambda_R \leq \lambda_R \otimes \gamma_{r,b} \Longleftrightarrow (\lambda_R \otimes \gamma_{r,b})(t) = \inf_{0 \leq s \leq t} \{\lambda_R(t-s) + \gamma_{r,b}(s)\}$$

$$\lambda_R \leq \{b + rt\} \wedge \{Rt\}$$

o que prova que o fluxo de entrada, A^{in}, é limitado (ou constrangido) pela função $\gamma_{r,b}$, como pressuposto inicialmente.

5.7.2 *Curva de serviço*

No caso da *curva de serviço* $\beta_{r,T}$ (com $T = 0$) obtém-se a curva de referência λ_r, uma vez que existe uma garantia de serviço de, pelo menos, o débito médio que as fontes deste tipo necessitam. A função escolhida para representar a *Curva de Serviço* foi λ_r, que demonstrou ser uma "boa" função de acordo com a Definição 8.

Este tipo de curvas oferecem uma forma interessante de visualizar o conceito de *Curva de Serviço*. Neste caso $\beta(u)$ é a quantidade de serviço mínimo garantido durante

o período em que a ligação está ocupada na sua totalidade. Como será observado, esta é a situação encontrada no problema em discussão, uma vez que a Definição 12 aplica-se directamente porque a *curva de serviço* foi definida *à priori*, $\beta(t) = r \times t$, logo a *curva de saída* é

$$B^{out}(t) \geq \inf_{s \leq t}\{A^{in}(s) + \beta(t-s)\} = \lambda_r,$$

o que prova que a *curva de serviço*, λ_r, é a definida como pressuposto inicial deste modelo e que é garantida quando o débito de saída do canal tem de ser equitativamente dividido por todos os fluxos que estão a consumir a largura de banda na totalidade; no entanto deve ser sempre verificada a condição

$$B^{out}(t) \geq \sum_{j=1}^{N} \lambda_r^j.$$

É ainda necessário testar se o modelo em discussão obedece ao Teorema 17, que indica que existe uma *curva de chegada* $\alpha^* = \alpha \oslash \beta$, que não é mais do que a curva que limita a curva de saída obtida. Como $\alpha = \gamma_{r,b}$ e $\beta = \lambda_r$, o resultado da operação de deconvolução é

$$(\gamma_{r,b} \oslash \lambda_r)(t) = \sup_{u \geq 0}\{\gamma_{r,b}(t+u) - \lambda_r(u)\} = \gamma_{r,b},$$

verificando-se que a *curva de chegada* que limita a curva de saída é a mesma que limita o fluxo de entrada e, como a *curva de saída* é sempre inferior ou igual à *curva de chegada*, fica justificada a escolha das *curvas de chegada* e de *serviço*.

5.7.3 Definição da *curva de saída*

A *curva de saída* (débito atribuído a cada fluxo), que permite que a distribuição dos atrasos e da ocupação dos *buffers* a jusante dos TBs seja equitativa, pode ser obtida através das ferramentas de modelação descritas. Assim, e com base no comportamento desejado relativamente aos débitos obtidos na distribuição da largura de banda pelos fluxos em competição, é possível obter a mesma representação gráfica através da combinação de algumas funções definidas inicialmente. A combinação, por exemplo, da função λ_R com a função $\beta_{R,T}$ e a utilização dos operadores $\wedge$ e $+$ da álgebra definida no Capítulo 3, permite obter novas expressões, incluindo a curva que representa, de forma generalizada, a *curva de saída*.

A *curva de saída* obtida tem significado para os casos apresentados na secção 5.5 (fontes *On/Off* com períodos bem definidos), aquando da definição dos vários cenários. A função que representa a *curva de saída* é definida pela expressão (A.3) para as N fontes com $j = 1, 2, ..., N$. Esta função desdobra-se para cada uma das curvas que representam o número de células servidas de cada um dos fluxos.

5.7.4 Relação dos parâmetros prospostos com as curvas

Basedo nos teoremas definidos por Le Boudec e R. Cruz relativamente aos limites dos atrasos e do *backlog* é possível realizar algumas extensões, com o objectivo de incluir e relacionar os novos parâmetros (X, Z, Q) introduzidos no modelo proposto.

A ideia de garantir um determinado débito a uma fonte (parâmetro $X_i \geq 0$) implica que a diferença entre a *curva de saída* $(y(t))$ e a *curva de serviço* $(\beta(t))$ seja sempre maior ou igual a zero $(X_i \geq 0)$; desta forma é sempre conhecida a quantidade de largura de banda que uma determinada fonte está a beneficiar acima do débito garantido, como indicado na Definição 18.

O número de *tokens* disponível no TB (parâmetro Z_i) tem um papel relevante na tomada de decisão relativamente à fonte que vai ser servida a seguir. Com este parâmetro é possível saber quantas células de um determinado *burst* já entraram no ED e foram processadas pelo TB associado. A sua importância revela-se em dois momentos: (i) quando uma fonte inicia a sua actividade (em rigor quando $Z \sim b$) e (ii) quando se verifica uma situação em que uma fonte inicia a sua actividade após ter sido já processado (passado pelo TB) mais de metade das células do *burst* da fonte que estava ser servida (em rigor quando $Z < b/2$). A segunda situação apresenta-se com uma importância relevante porque se um dos objectivos é realizar uma distribuição equitativa dos atrasos e o controlo da ocupação dos *buffers*, é necessário ser tratada em separado (ver secção 5.5 Cenário 2). Com base neste pressuposto é possível conhecer a evolução dos *bursts*, através da diferença entre a *curva de chegada* $(\alpha(t))$ e a curva do fluxo de entrada $(x(t))$, como indicado na Definição 19.

O último dos três parâmetros (Q_i) pode ser obtido e limitado superiormente pelo valor total do *bucket* (b). Esta situação só é possível devido ao paralelismo das *curvas de chegada* e de *serviço*, sendo possível verificar que o limite máximo de ocupação da fila a montante do TB nunca será superior ao tamanho máximo do *bucket* (teorema 20).

5.8 Parametrização do método

A partir do momento em que a *curva de saída* para cada fluxo i (com $i = 1, 2, ..., N$) está definida matematicamente, é necessário parametrizar o modelo para a sua justificação.

$A_i^{in}(t)$	é o número de células que chegam ao *token bucket* i, durante o instante de tempo t;
$A_i^{in}(s,t)$	é o número de células que chegam ao *token bucket* i, durante o intervalo de tempo $[s,t]$ com $t \geq s$;
$B_i^{out}(t)$	é o número de células que são servidas do *buffer* a jusante do *token bucket* i, no instante de tempo t;
$B_i^{out}(s,t)$	é o número de células que são servidas do *buffer* a jusante do *token bucket* i, durante o intervalo de tempo $[s,t]$ com $t \geq s$;
$B^{out}(t)$	é o número máximo de células servidas pelo escalonador, no instante de tempo t;
C	é a capacidade do canal de saída;
$C_i(t)$	representa o número de células transmitidas pela fluxo i, no instante de tempo t;
$C_i(s,t)$	representa o número de células transmitidas pela fluxo i, no intervalo $[s,t]$, com $t \geq s$;
$C_{min}(t)$	representa o número mínimo de células transmitidas, no instante de tempo t que serve de referência ao fluxo que está a ser servido;
$X_i(t)$	representa o número de *tokens* de transmissão que foram distribuídos em excesso, em relação ao débito médio garantido do *token bucket* i, durante o instante de tempo t;
$Z_i(t)$	representa o número de *tokens* disponíveis no *token bucket* i, durante o instante de tempo t;
$Q_i(t)$	é o número de células no *buffer* a jusante do *token bucket* i durante o instante de tempo t;
b_i	é o tamanho máximo do *bucket* i;
t_p	representa o tempo que demora a processar $C_{min}(t)$ células ao débito $R^{out}(t)$;
$tFlag_i$	parâmetro que indica se o fluxo i já foi servido pela primeira vez;
$Trans_i$	parâmetro que indica se o fluxo i está a ser servido ou se está aguardar.

Figura 5.8: Definição dos parâmetros para a parametrização do método.

Para a parametrização é necessário assumir que o escalonador, por um lado, recebe células de N fluxos e serve tantas quantas a sua capacidade o permitir (esta capacidade não é constante porque estes fluxos são agregados e mapeados num VC do tipo ABR) e, por outro lado, para cada conexão garante a *curva de serviço* $\beta_i(t)$ (neste caso $\beta_i(t) = SCR_i \times t$, para todo o $t > 0$), com o objectivo de fornecer um débito de saída nunca inferior a $\beta_i(t)$. A par deste pressuposto é necessário definir o conjunto

de parâmetros (ver Figura 5.8) que vão ser utilizados na justificação do método de escalonamento proposto.

Com base nas definições estabelecidas por Cruz [102, 27] e Le Boudec [12, 11, 13] e nos parâmetros definidos para o modelo é possível definir a política de escalonamento utilizada por este método. O objectivo é encontrar um algoritmo de escalonamento que, com base nas *curvas de chegada* e de *serviço* para uma dada conexão e nas Definições 18, 19 e no Teorema 20 permita encontrar uma *curva de saída* (apresentada na secção anterior), de forma a obter vantagem da multiplexagem estatística, mas tendo sempre presente as seguintes condições

$$B_i^{out}(t) \geq \beta_i(t) \tag{5.13}$$

e

$$\sum_{i=1}^{N} B_i^{out}(t) = B^{out}(t) \leq C. \tag{5.14}$$

Definindo $\psi(t)$ como a função que para cada t permite obter o instante de tempo em que o servidor passou a servir outra fonte, terá de ser

$$\psi(t) = max\{s : s \leq t \text{ e } C_i(s) = C_{min}(s)\} \text{ para } i = 1, ..., N \tag{5.15}$$

e assim é possível afirmar que o servidor garante uma *curva de saída* $B_i^{out}(t)$ para a conexão i, tal que

$$B_i^{out}[\psi(t), t] = \begin{cases} B^{out}(t), \text{ se } C_i(t) \leq C_{min}(t) \\ 0, \text{ se } C_i(t) = 0 \end{cases} \tag{5.16}$$

Sendo $\xi_i(t)$ o tempo de chegada da célula que se encontra à cabeça do *buffer* i e $F(t)$ a fonte que está a ser servida no instante t, então é possível saber qual a fonte que será servida assim que se verifique a condição $C_i(t) = C_{min}(t)$, com o tempo necessário para transmitir $C_{min}(t)$ dado por

$$t_p = \frac{C_{min}(t) \times 424}{C}, \tag{5.17}$$

em que 424 representa o tamanho da célula em *bits*, logo,

$$F(\psi(t) + t_p) = min\{\xi_i(t)\} \text{ para } i = 1, 2, ..., N. \tag{5.18}$$

Uma vez terminada a parametrização geral do modelo, tratam-se os casos apresentados na secção 5.5.

1^o caso

Admitindo que a próxima fonte será servida no instante de tempo $I = \psi(t) + t_p$, desde que nenhuma outra fonte inicie a sua actividade antes de I, e não existindo qualquer alteração do estado (nenhuma outra fonte inicia a sua actividade ou porque deixa de transmitir ou porque reduz significativamente o seu débito), é possível dizer que a partir do momento em que se verifique a condição $C_i(t) = C_{min}(t)$, e admitindo que está a ser servida a fonte i, a próxima fonte a ser servida em I é

$$F(I) = min\{\xi_i(I)\} \text{ para } i = 1, 2, ..., N \tag{5.19}$$

desde que

$$X_i(I) \geq 0 \text{ para } i = 1, 2, ..., N$$

2^o caso

Considerando a possibilidade de uma fonte iniciar a sua actividade em qualquer instante de tempo (antes de I) em que a condição $C_i(t) = C_{min}(t)$ ainda não se verifica, é necessário, por um lado, definir uma nova condição que permita alterar a fonte que está a ser servida e, por outro lado calcular o novo número de células a transmitir e que serve de referência ($C_{min}(t)$). Como estabelecido na Definição 19, o parâmetro $Z_i(t)$ indica o estado em que se encontra o *burst* e como esse parâmetro é inicializado com b_i ($Z_i(0) = b_i$), então será escolhido o fluxo que apresenta o maior valor, mas desde que nunca tenha sido servido. A condição que está subjacente à expressão "*nunca tenha sido servido*" deve-se ao facto da fonte que inicia a sua actividade poder ter um valor de $Z_i(t)$ inferior à fonte que está a ser servida. Se esta condição não fosse considerada, assim que a segunda fonte iniciasse a sua actividade seria a primeira que continuaria a beneficiar do serviço do escalonador e desde logo o problema da equidade da distribuição dos atrasos não seria resolvido. Para verificar se a fonte já foi alguma vez servida considerou-se o parâmetro $tFlag_i$, inicializado a zero no instante em que o sistema está vazio, isto é, quando $Q_i(t) = 0$ e o número de células transmitidas é igual ao número total de células do *burst*.

Desta forma a fonte que será servida é obtida de modo diferente do *1º caso*, por isso, é necessário conhecer o instante em que uma fonte entra em actividade. Definindo $\eta(t)$ como o último instante de tempo antes de t em que uma nova fonte iniciou a sua actividade, tem-se

$$\eta(t) = max\{s : s \leq t \text{ e } Z_i(s)\} \text{ para } i = 1, 2, ..., N, \tag{5.20}$$

desde que

$$tFlag_i = 0, \text{ para } i = 1, 2, ..., N$$

e o número de células transmitidas (valor de referência) que a nova fonte tem disponível para transmitir é

$$C_{min}(t) = C_i(\eta(t)), \tag{5.21}$$

uma vez que se verifica a condição $C_i(t) < C_{min}(t)$. Assim, o fluxo que estava a ser servido tem de aguardar uma nova oportunidade e desta forma é necessário indicar que a partir do instante em que se verifica a entrada da nova fonte em actividade, o fluxo que está a ser servido fica em repouso, isto é, $tFlag_i = 1$.

3º caso

O terceiro caso apresenta-se como uma excepção que necessita de ter um tratamento diferente. Com efeito, quando uma segunda fonte inicia a sua actividade após ter sido já processado no ED mais de metade das células do *burst* da primeira, o cálculo do número de células a transmitir pela segunda fonte é diferente dos processos apresentados até ao momento. Para resolver este problema é utilizada a informação fornecida por $Z_i(t)$ e b_i para saber se um determinado *burst* já transmitiu células suficientes para servir metade das suas células, assim

$$\nu = \frac{Z_i(t)}{b_i}. \tag{5.22}$$

Utilizando a Definição 19 com $Z_i(t) = \alpha(t) - x(t)$ e o resultado de (5.22) é possível verificar se $\nu < 0.5$ já foi atingido. Desta forma consegue-se saber com precisão o número de células a transmitir ainda necessárias para servir o resto do *burst*, recorrendo ao tamanho máximo do *burst* indicado ao ED e à função $\eta(t)$, de acordo com

$$C_{min}(t) = MBS_i - C_i(\eta(t)). \tag{5.23}$$

Para além dos três casos considerados existe um que se sobrepõe a todos os outros, isto é, sempre que se verifique a condição $X_i(t) < 0$ (condição que indica que $R_i^{out}(t) < \beta_i(t)$), mesmo com $C_i(t) < C_{min}(t)$ é retirado o serviço à fonte que estava a beneficiar do recurso e o serviço será disponibilizado à fonte que se encontra nessa situação ($X_i(t) < 0$), para que seja sempre possível garantir o débito médio garantido como estabelecido na Definição 18.

Em todos os casos é necessário considerar a situação em que exista de facto uma mudança relativamente à fonte que estava a ser servida. Assim, quando se verifica uma das condições definidas anteriormente, por exemplo se a fonte a ser servida é $F(I)$, então a indicação de serviço passa para zero enquanto a que estava a ser servida passa para um, isto é, $Trans_{F(I)} = 0$ e $Trans_{F(t)} = 1$.

5.9 Conclusões

Como verificado nos métodos de escalonamento apresentados na literatura, o objectivo principal é servir pacotes ou células de forma a que os limites máximo de atraso não sejam excedidos. De acordo com os resultados de simulação, os objectivos definidos na tese não são atingidos com os modelos de escalonamento FIFO e WFQ (ver Anexo C), isto é, os atrasos não são distribuídos equitativamente. Foi precisamente este problema que motivou a proposta de um novo método de escalonamento.

A solução encontrada envolveu a introdução de TBs nas portas de entrada dos EDs para modelar/policiar o tráfego e com base na sua informação fazer escalonamento de acordo com os objectivos definidos inicialmente. Para atingir estes objectivos utilizou-se uma *curva de chegada* ($\alpha_i(t)$) que limita superiormente os fluxos de entrada e uma *curva de serviço* ($\beta_i(t)$) que limita o serviço inferiormente. As curvas escolhidas obedecem aos critérios estabelecidos pela ferramenta matemática *Network Calculus* [12] como sendo curvas "boas", o que se veio a verificar, como pode ser comprovado no Anexo A, secção A.2. Após a escolha deste tipo de curvas e da decisão em obter *curvas de saída* (para um determinado tipo de fontes) idênticas às das Figuras 5.4 (b) e 5.5 (b), isto é, atribuição de períodos em que o escalonador disponbiliza a largura de banda na sua

totalidade, foi possível definir matematicamente a *curva de saída* para cada um dos fluxos em competição.

Estando as curvas definidas matematicamente foi necessário parametrizar o problema de forma a resolver as situações definidas na secção 5.5, podendo assim concluir-se que para um fluxo de tráfego é atribuído um conjunto de períodos de transmissão ao débito máximo de saída, em que cada período é composto por um determinado número de células a transmitir ($C_{min}(t)$) de forma a que **o atraso seja distribuído equitativamente (de acordo com os objectivos definidos na secção 5.5) e a ocupação dos *buffers* controlada entre todos os fluxos em competição.**

Assim os objectivos definidos inicialmente são realizáveis para um dado conjunto de fluxos servidos por um escalonador com capacidade $B^{out}(t)$ e com $\sum_{i=1}^{N} B_i^{out}(t) \leq B^{out}(t)$, para qualquer valor de $t > 0$, quando a *curva de chegada* é $\alpha_i(t)$, a de *serviço* é $\beta_i(t)$ e a *curva de saída* $B_i^{out}(t) \geq \beta_i(t)$ para $i = 1, 2, ..., N$, desde que qualquer situação que ocorra possa ser incluida nos cenários considerados.

BIBLIOTECA F.E. PORTO

Capítulo 6

Algoritmos para escalonamento de tráfego

Neste capítulo é apresentado um conjunto de soluções algorítmicas utilizadas para implementar o modelo apresentado no Capítulo 5. Estas soluções começaram com a adaptação de um algoritmo descrito na literatura, por forma a obter os primeiros resultados e evoluiram para soluções próprias e concebidas de raiz. Estas soluções serão analisadas através de simulação em seis cenários definidos no Capítulo 5. Como será observado, à medida que uma nova versão é apresentada, os resultados melhoram relativamente à versão anterior; no entanto, a complexidade dos algoritmos vai também crescendo, porque os problemas detectados em cada versão, tendo sempre presente os casos apresentados na secção 5.5 do Capítulo 5, assim o obrigam.

Na primeira secção é apresentada uma solução algorítmica que serviu como ponto de partida para a obtenção de resultados em relação ao método proposto [84]. Na secção seguinte é introduzida uma solução intermédia, onde é apresentada uma ideia interessante do ponto de vista da implementação devido à sua simpliciade de processos, e que serviu de base para a introdução de um novo conceito que não tinha sido explorado na solução anterior, mas foi aplicada na segunda solução algorítmica. Na terceira secção é discutida a última versão algorítmica que utiliza o conceito introduzido anteriormente, mas com uma variante em relação à forma de cálculo, para além da sua avaliação através dos resultados obtidos por simulação. Na quarta secção é discutida a complexidade algorítmica das três versões algorítmicas apresentadas. Na última secção são extraídas algumas conclusões enquadradas no resumo do trabalho apresentado no presente capítulo.

6.1 Primeira solução

A primeira solução algorítmica serviu como ponto de partida para a obtenção de resultados relativamente ao método proposto, com o objectivo de perceber se as opções tomadas em relação aos parâmetros derivados foram acertadas, para além de obter pontos de referência para o melhoramento das soluções seguintes.

6.1.1 Solução algorítmica

Para demonstrar a utilidade do método apresentado no Capítulo 5 foi considerado inicialmente, devido à sua simplicidade, o algoritmo *Least Time to Reach Bound* (LTRB) [8], com adaptações inerentes aos objectivos identificados. Este algoritmo foi adaptado a fluxos regulados, ou conformes, isto é, manteve-se a seguinte ideia: o fluxo seguinte a ser escalonado é aquele cuja fila de espera irá saturar primeiro, sob uma circunstância hipotética de que o débito de entrada seja o máximo possível (isto é, limitado por $\gamma_{r,b}$) e o débito de saída o mínimo possível que garanta o débito médio. De acordo com a análise efectuada no capítulo precedente foi considerado que a saturação só será atingida quando $Q_i = b_i$ (desde que este seja o pior caso sob uma distribuição fixa de SCR_i). Além disso, uma vez que uma fonte é conforme, o TB não pode transmitir com PCR_i permanentemente. Por isso, foi introduzida a seguinte limitação: sempre que $Z_i = 0$, a fonte continuará a transmitir, mas desta vez com SCR_i; similarmente, antes que a saturação seja alcançada, é imposto um débito de saída zero (o mínimo possível), mas mais tarde deve ter um valor tal que o nível de saturação não seja excedido.

Considerando o estado actual de um fluxo (Z_i, X_i, Q_i) e impondo as condições discutidas, existem dois casos de interesse que devem ser analisados. No primeiro caso, se a situação $X_i = 0$ (a transmitir a SCR_i) for atingida primeiro, a saturação é alcançada após T_i^1, tal que

$$T_i^1 = \frac{X_i}{SCR_i}. \tag{6.1}$$

Por outro lado, se ocorrer primeiro $Z_i = 0$ (o *burst* foi completamente recebido), o instante em que se verifica a saturação é T_i^2, tal que

$$T_i^2 = \frac{b_i - Q_i}{PCR_i} \tag{6.2}$$

(quando a saturação for alcançada, $Z_i > 0$ e $X_i = -Z_i$). Consequentemente,

$$T_i^t = max(T_i^1, T_i^2) \tag{6.3}$$

é o tempo necessário para saturar a fila i; de acordo com o algoritmo adaptado do LTRB, o fluxo a ser escalonado é o que tem o tempo mínimo de saturação entre todos os fluxos que podem ser escalonados.

/* Inicialização */

01. $T_t = 0$

02. $MinTime = 99999$

/* Escolher a fonte a ser escalonada */

03. $for\ \forall_i$(fila não vazia)

04. $\quad x_i \leftarrow b_i - (Q_i + z_i)$

05. $\quad T_i^1 \leftarrow x_i / SCR_i$

06. $\quad T_i^2 \leftarrow (b_i - Q_i) / PCR_i$

07. $\quad T_t \leftarrow MAX(T_i^1, T_i^2)$

08. $\quad if\ (MinTime > T_t)$

09. $\quad\quad MinTime \leftarrow T_t$

10. $\quad\quad Fonte \leftarrow i$

/* Escalonar uma célula da fila Fonte */

11. $Q_{ABR} \leftarrow Q_{Fonte}$

6.1.2 Resultados de simulação

O algoritmo descrito foi testado através de simulação, com o objectivo de avaliar os méritos do escalonamento dos fluxos IP. Foram considerados seis cenários (ver secção 5.5 do Capítulo 5) e foram analisados três parâmetros: a ocupação das filas a jusante dos TBs, o atraso das células e o valor do parâmetro X para cada uma das fontes.

Nas simulações descritas nesta secção, apenas um canal de largura de banda fixa foi considerado com um débito ligeiramente acima (10%) do débito médio total do fluxo

agregado, de forma a garantir que não existe acumulação de tráfego para os períodos seguintes.

Cenário 1

Os resultados obtidos com o Cenário 1 são pouco conclusivos porque o decalamento entre as fontes é reduzido, para além de que permitem identificar apenas alguns dos problemas que podem estar associados a esta solução.

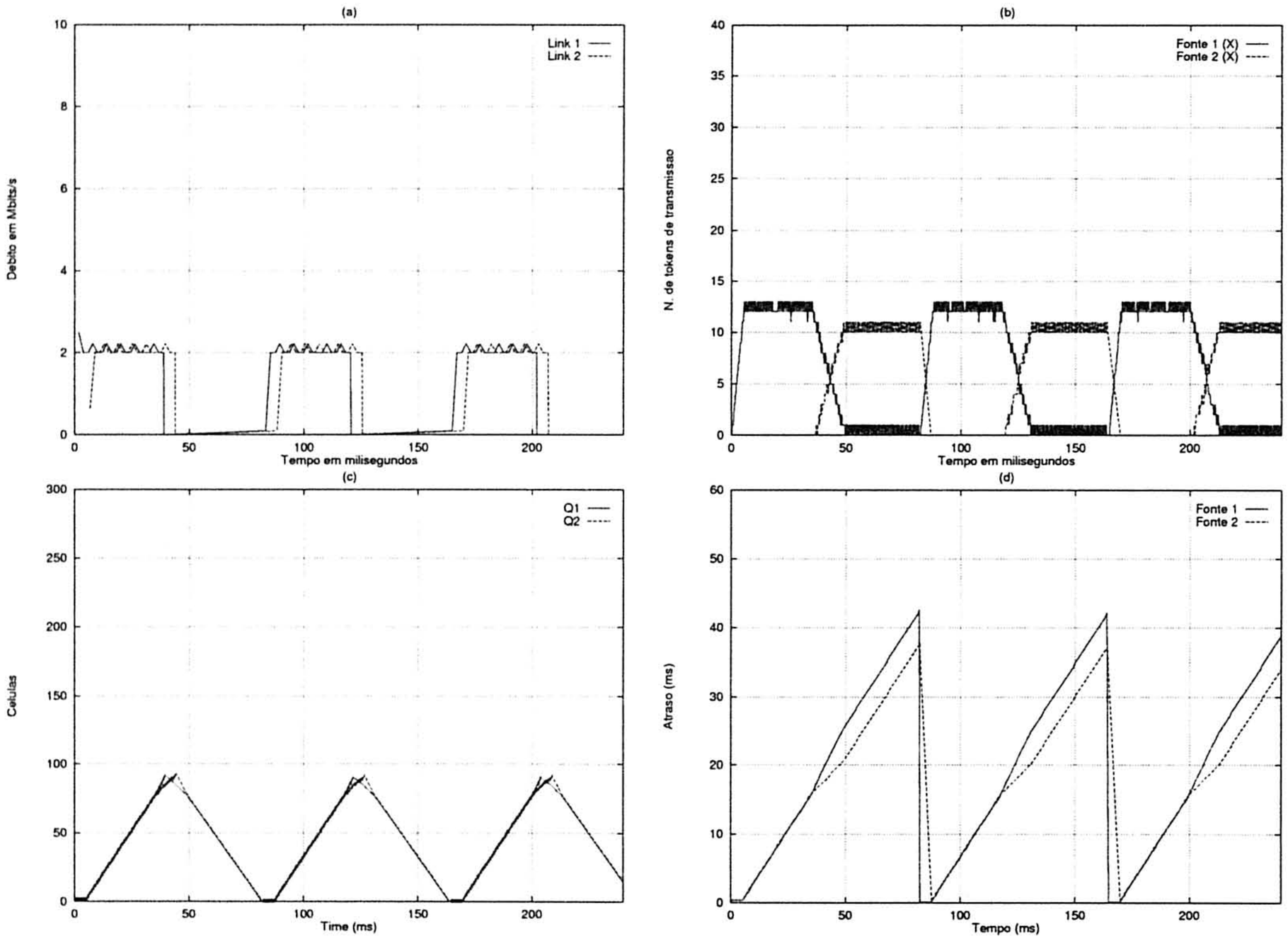

Figura 6.1: Resultados obtidos com o primeiro algoritmo no Cenário 1.

Nos resultados apresentados na Figura 6.1 é possível observar que a ocupação dos *buffers* das fontes (gráfico (c)) é coincidente, o que implica uma distribuição igual da largura de banda pelas duas fontes sem ter em consideração o tempo de início de cada uma delas, o que mostra um controlo da ocupação realizado de forma correcta, uma vez que em nenhum dos casos foi alcançado a ocupação máxima. A distribuição dos atrasos (gráfico (d)) já não se efectua da mesma forma e penaliza a fonte que inicia a sua actividade mais cedo, porque as que iniciam a sua actividade mais tarde vão beneficiar de uma maior largura de banda no fim dos seus períodos de actividade.

Como forma complementar, os valores do parâmetro X (gráfico (b)) mostraram

o modo mais ou menos eficiente como a largura de banda é distribuída dentro do ED. Através da observação do gráfico (*b*) é possível comprovar a afirmação produzida anteriormente porque a fonte que começa primeiro vai ter inicialmente mais largura de banda e assim a ocupação da sua fila é menor; no entanto, a que começa mais tarde vai ter mais largura de banda no fim do seu período de actividade.

Cenário 2

Os resultados obtidos com o Cenário 2 servem para estudar a distribuição dos atrasos e a ocupação dos *buffers*, quando uma fonte está quase no fim do seu *burst* e uma segunda começa a transmitir. O objectivo desta simulação é estudar se a disciplina de serviço utilizada toma em consideração esta situação, ou se apenas começa a receber células da segunda fonte e distribui a largura de banda equitativamente pelas duas.

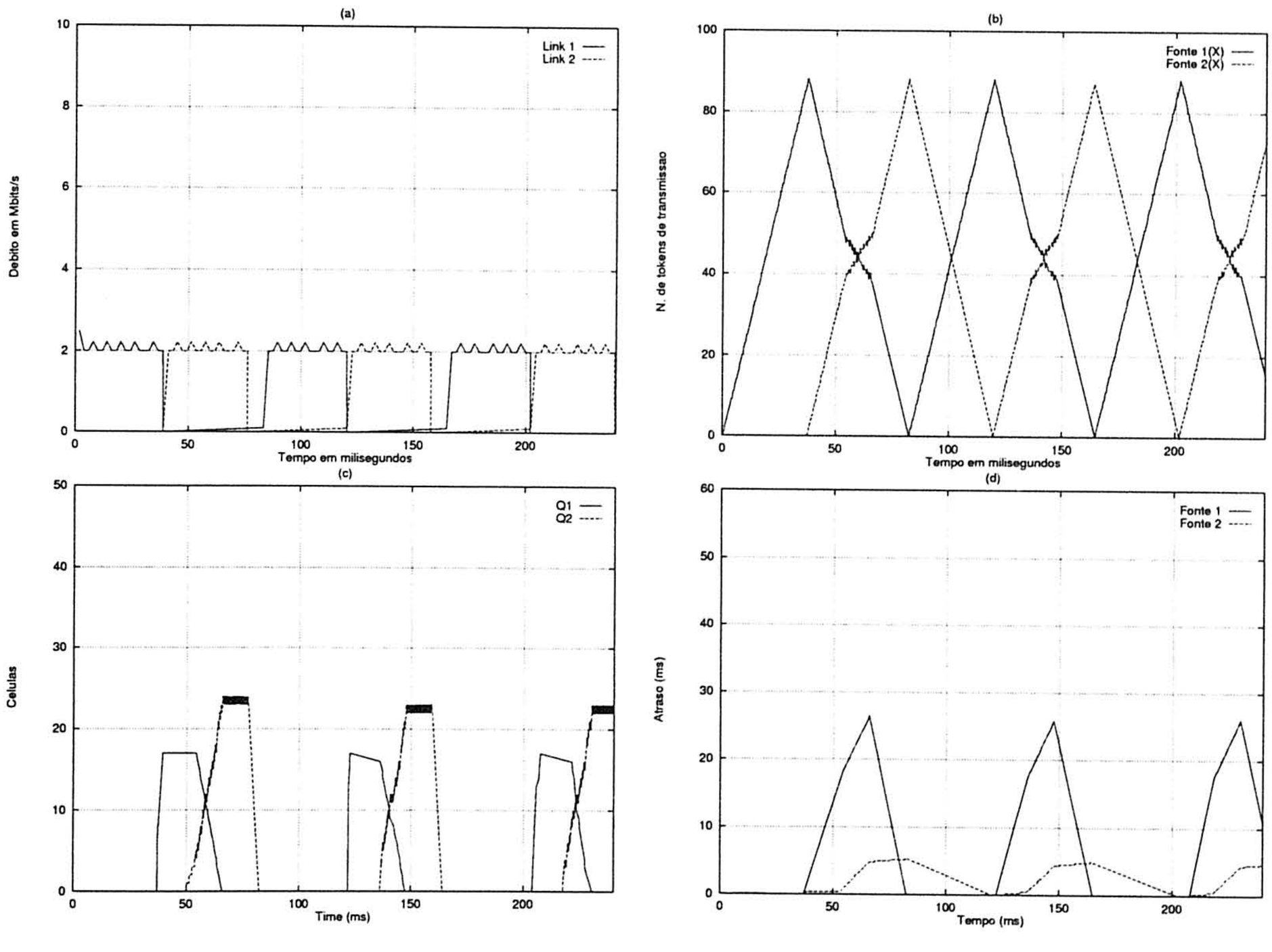

Figura 6.2: Resultados obtidos com o primeiro algoritmo no Cenário 2.

A distribuição dos atrasos é muito pouco equitativa, como pode também ser comprovado pela ocupação dos *buffers* a jusante dos TBs (resultados apresentados na Figura 6.2 (c)). O período de tempo que a primeira fonte tem de esperar para voltar a ser servida é demasiado grande quando comparado com o tempo de decalagem que é de

37.4 ms. Mais uma vez a ocupação máxima não foi atingida, mas a fonte que inicia a sua actividade mais tarde vai beneficiar de uma maior distribuição da largura de banda, assim que as células da primeira fonte sejam completamente servidas.

Da mesma forma que para o Cenário 1, os valores de X (Figura 6.2 (b)) servem para complementar a explicação e mostrar que para as duas fontes o momento crítico acontece no instante em que a segunda fonte inicia a sua actividade e recebe de imediato toda a largura de banda; sendo apenas no instante em que os valores de X se igualam que a primeira fonte volta a ser servida. Este facto leva a um aumento significativo dos atrasos impostos à primeira fonte.

Cenário 3

Os resultados obtidos com o Cenário 3 servem para estudar uma situação simples, mas que permite tirar algumas conclusões sobre a operação da disciplina de serviço utilizada. O objectivo desta simulação é verificar qual o comportamento do algoritmo de escalonamento quando existem três fontes a transmitir, mas uma começa a transmitir no início da simulação e as outras duas iniciam em simultâneo a sua actividade, mas decaladas da primeira de 10 ms. Neste cenário aparecem duas situações com interesse para análise, ou seja, o tempo de início entre a primeira e as outras duas e o tempo nulo de início entre as duas últimas.

Os resultados da Figura 6.3 vêm confirmar os problemas detectados no cenário anterior, isto é, as fontes que iniciam a sua actividade mais tarde acabam por tirar partido desta solução, porque assim que iniciam a actividade são servidas durante bastante mais tempo do que deviam, o que se verifica no gráfico (c). Neste gráfico é possível observar a ocupação dos *buffers* colocados a jusante dos TBs e, logo que as duas fontes coincidentes iniciam a sua actividade, a ocupação da primeira cresce muito rapidamente, enquanto que a ocupação das outras duas tem um crescimento mais lento. E ainda possível observar que a drenagem do *buffer* da primeira fonte é realizada de forma mais lenta do que das outras duas, reflectindo-se assim na distribuição dos atrasos (gráfico (d)).

Os valores de X apresentados no gráfico (b) da Figura 6.3 permitem observar que, após as duas fontes coincidentes iniciarem a sua actividade, a largura de banda é-lhes atribuída em detrimento da primeira (parecendo até que o algoritmo está a comportar-se de acordo com os objectivos definidos) e, é precisamente nos instantes em que os valores

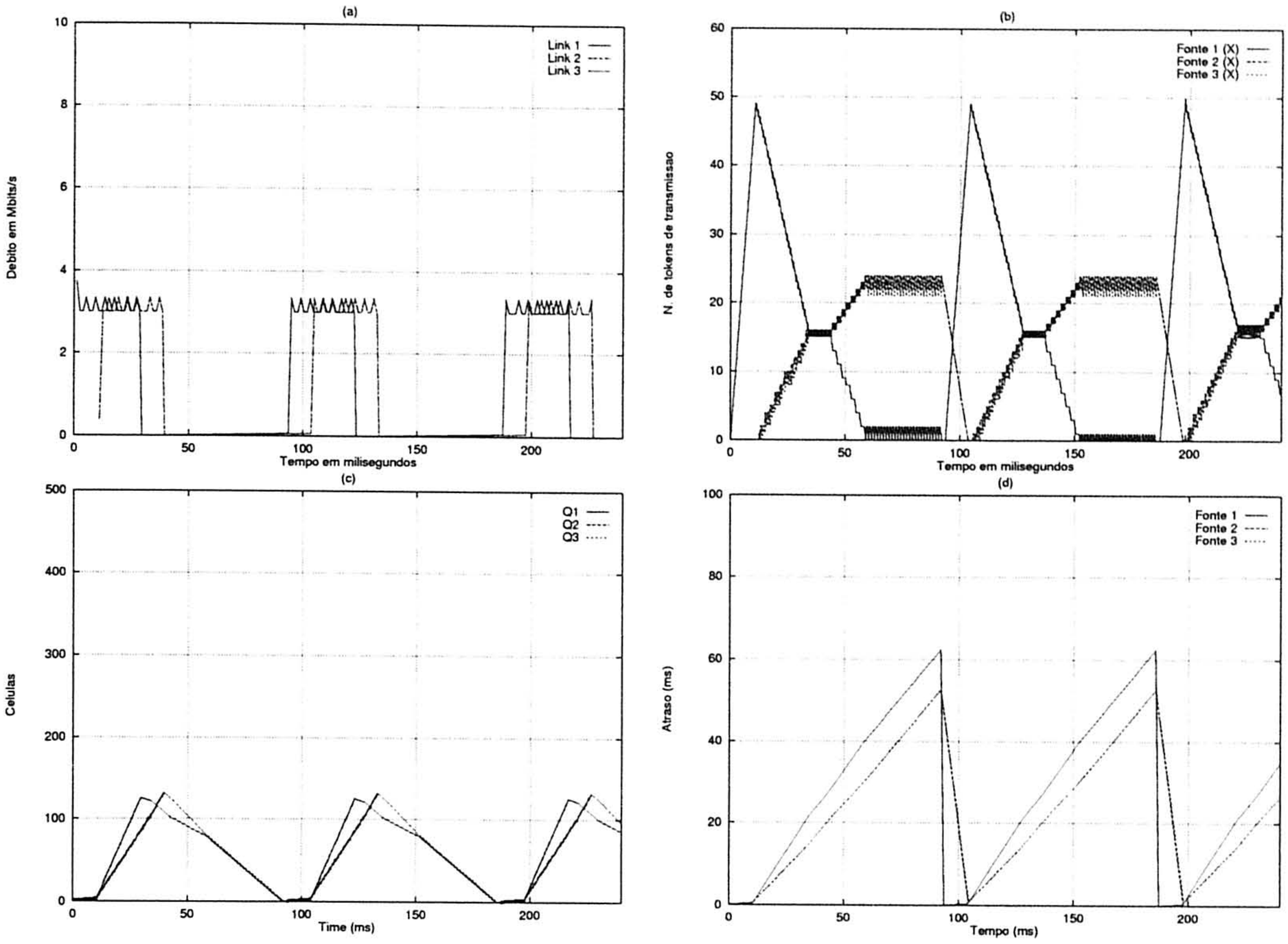

Figura 6.3: Resultados obtidos com o primeiro algoritmo no Cenário 3.

de X se cruzam que a primeira fonte volta a ser servida, o que vai levar a uma falta de equidade na distribuição dos atrasos.

Cenário 4

Os resultados obtidos com o Cenário 4 serviram para estudar uma situação análoga à anterior, mas com os dados distribuídos de forma diferente, em vez de duas fontes a transmitir em simultâneo, apenas existe uma com os parâmetros atribuídos com o dobro dos valores. O objectivo é perceber, por um lado se a disciplina de serviço atribui 1/3 da largura de banda à primeira fonte e 2/3 à segunda e, por outro lado perceber se existem problemas de equidade na distribuição dos atrasos.

É possível verificar, através dos resultados apresentados na Figura 6.4, uma distribuição mais razoável da largura de banda pelas fontes em competição e, essencialmente as ocupações dos *buffers* e a distribuição dos atrasos parecem mais coincidentes com os requisitos iniciais. Relativamente à ocupação pode-se verificar que esta é mais do dobro da segunda fonte em relação à primeira; no entanto os atrasos já não reflectem directamente essa ocupação. O problema está na forma como a largura de banda é atribuída

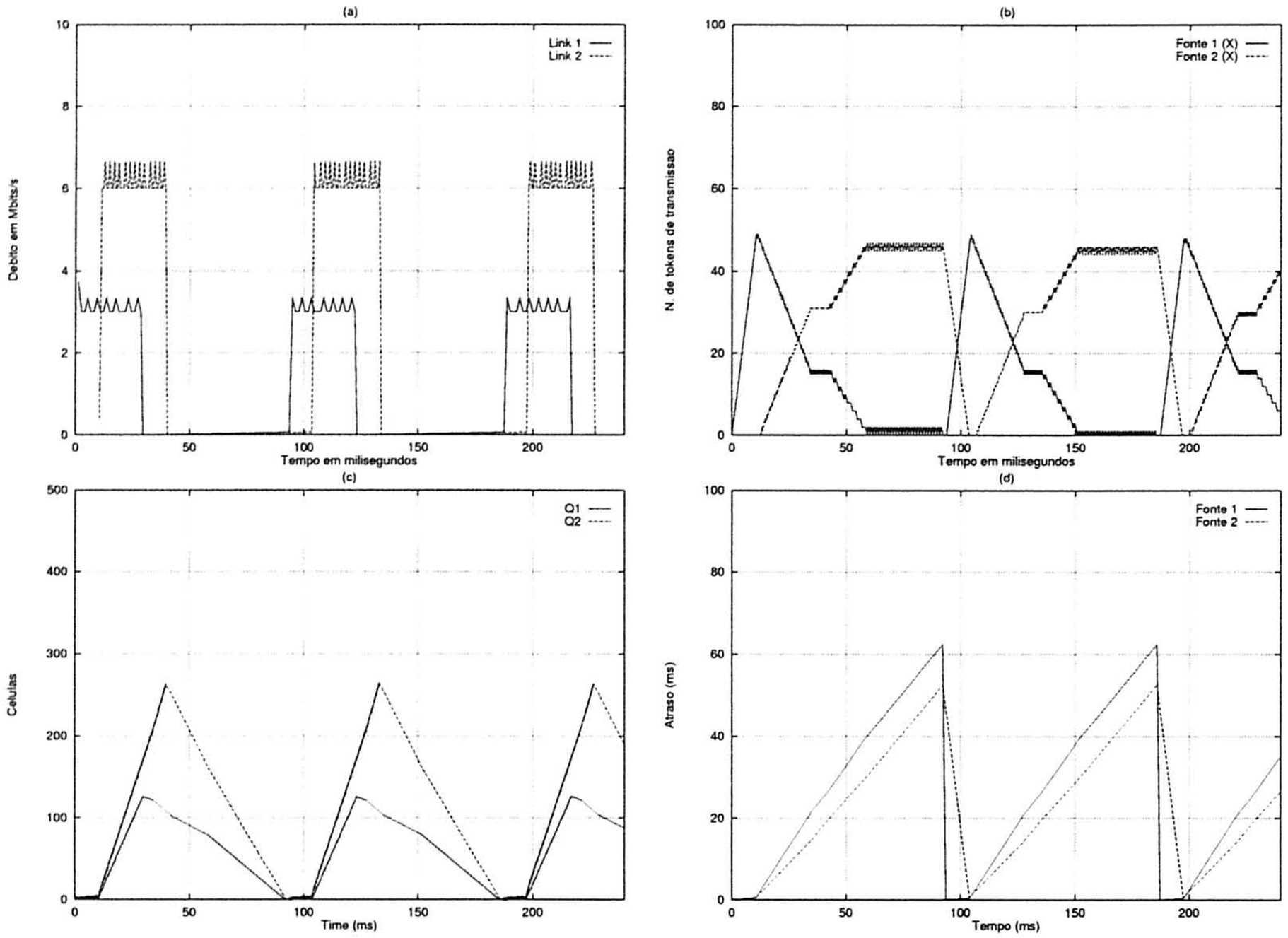

Figura 6.4: Resultados obtidos com o primeiro algoritmo no Cenário 4.

e a drenagem dos *buffers* é bem exemplo disso; basta observar que o *buffer* da segunda fonte (gráfico (c)) tem uma drenagem muito mais rápida do que o da primeira e é esse facto que vai fazer com que a distribuição dos atrasos não seja equitativa.

Com o mesmo objectivo dos resultados apresentados para o cenário anterior, os valores de X para as duas fontes explicam qual o problema, porque basta observar o gráfico (b) e verificar que a Fonte 1 está sem ser servida, sensivelmente durante 30 ms e depois a banda é partilhada entre as fontes, o que leva aos resultados observados na distribuição dos atrasos.

Cenário 5

Os resultados obtidos com o Cenário 5 têm como objectivo estudar a reacção da disciplina de serviço ao problema da escalabilidade, considerando quatro fontes; neste caso o início das fontes é decalado de 5 ms. Neste tipo de cenário é fácil de perceber a forma como a distribuição dos atrasos é realizada e se as fontes que iniciaram a sua actividade mais cedo são ou não beneficiadas.

Os problemas já detectados nos resultados obtidos com os cenários precedentes

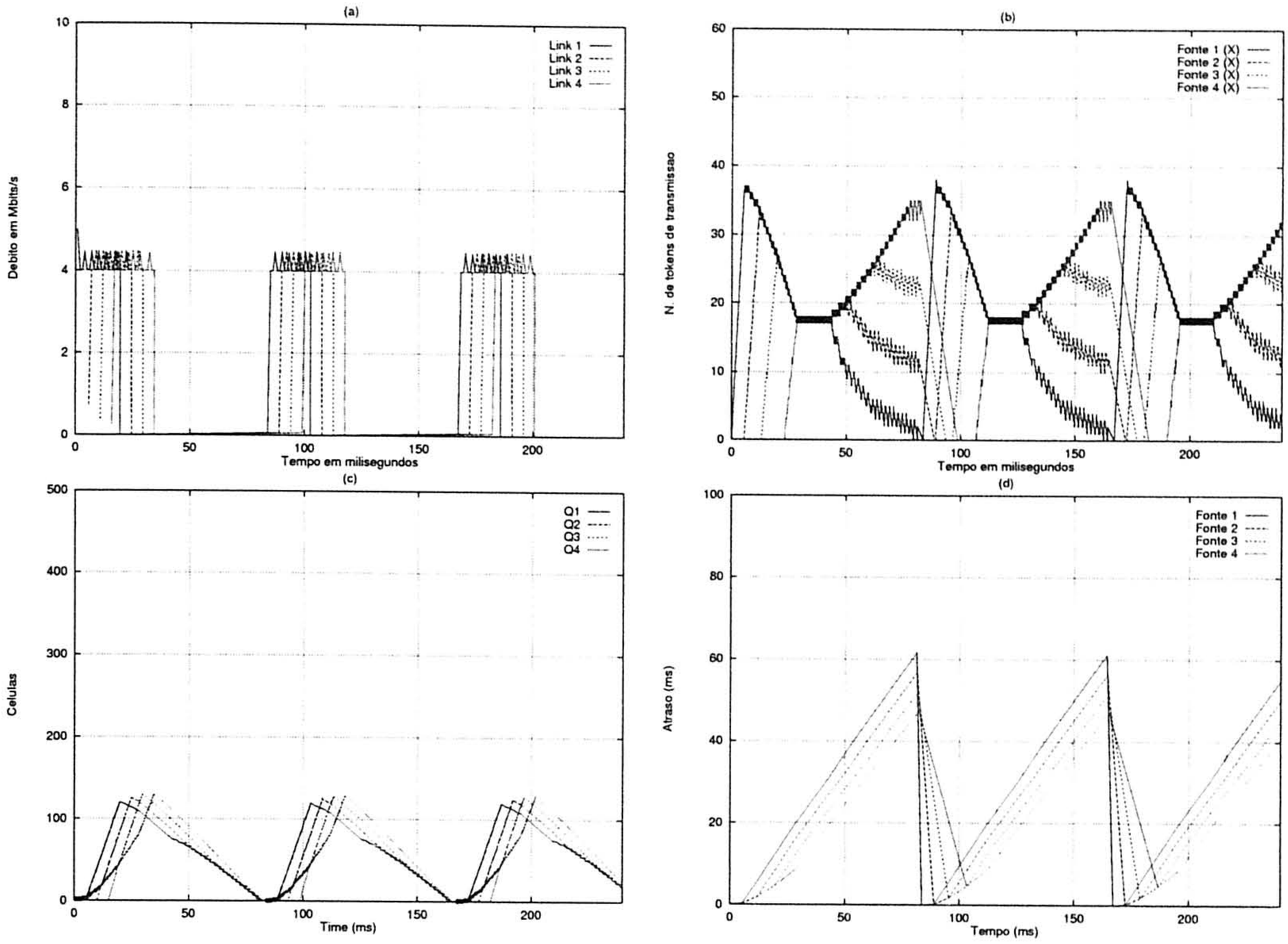

Figura 6.5: Resultados obtidos com o primeiro algoritmo no Cenário 5.

são confirmados pelos resultados apresentados na Figura 6.5; por exemplo, a ocupação dos *buffers* e a distribuição dos atrasos neste cenário mostram que as fontes que iniciam a sua actividade mais tarde acabam por ser beneficiadas, porque basta observar a ocupação do *buffer* da Fonte 1 (gráfico (c)), para verificar que em vez de ficar vazio em primeiro lugar acaba por ser um dos últimos, o que vai ter implicações directas nos maiores atrasos verificados.

Os valores de X observados no gráfico (b) da Figura 6.5 são bastante elucidativos para a explicação dos resultados obtidos com este cenário, dada a tendência que o algoritmo tem em produzir valores idênticos de X, leva à distribuição equitativa da largura de banda, mas ao fazé-lo deixa de distribuir equitativamente os atraso e, consequentemente não consegue atingir os objectivos propostos inicialmente.

Cenário 6

Os resultados obtidos com o Cenário 6 servem para confirmar os problemas encontrados nos resultados obtidos com o cenário anterior. O objectivo desta simulação é o mesmo da anterior, mas aumentando os tempos de início entre as fontes de 5 para 10 ms.

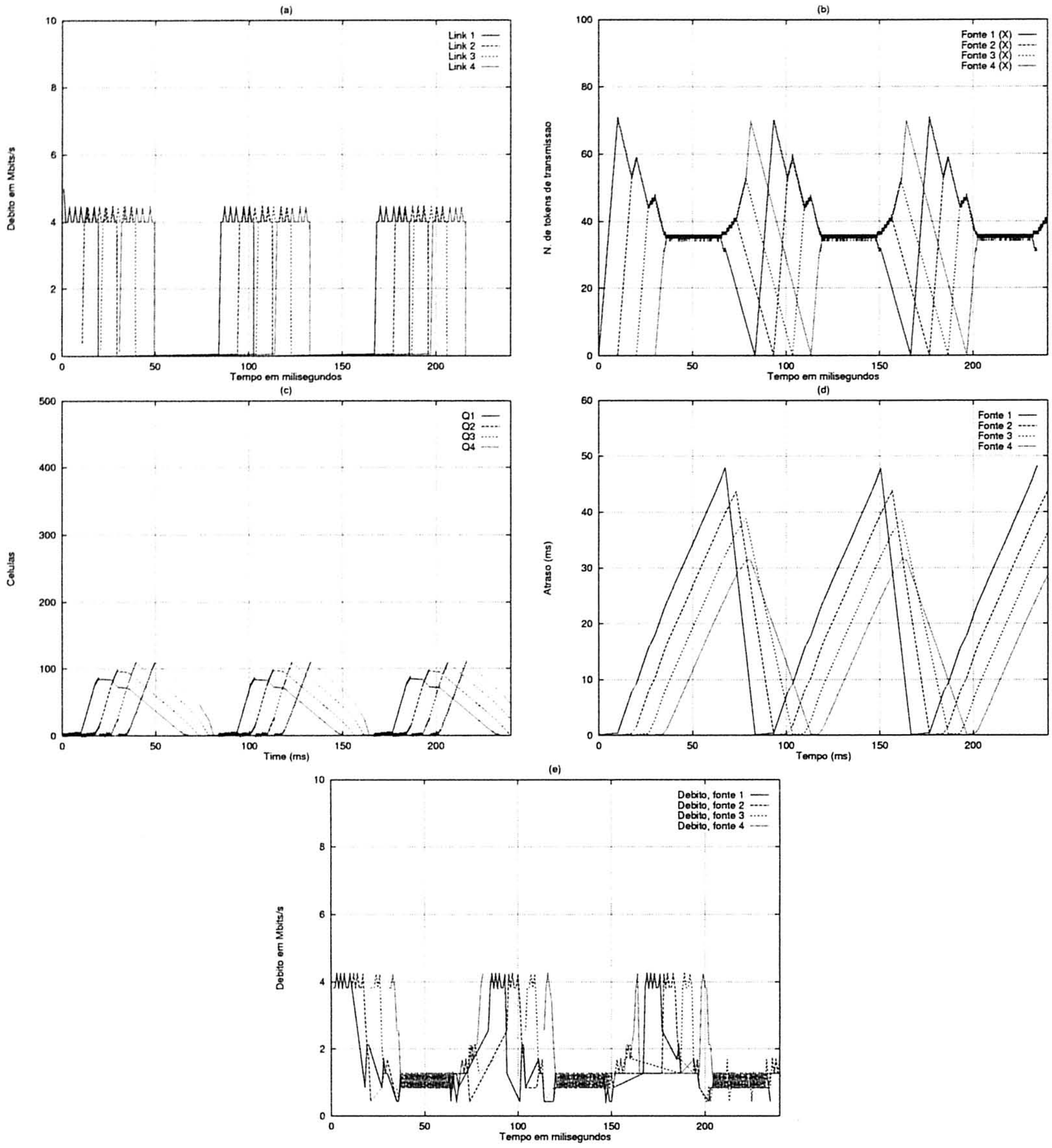

Figura 6.6: Resultados obtidos com o primeiro algoritmo no Cenário 6.

Todas as justificações dadas até ao momento podem ser confirmadas através dos resultados da Figura 6.6, que mostram de forma muito clara a "injustiça" na distribuição dos atrasos. Basta uma simples observação da ocupação dos *buffers* (gráfico (d)) para verificar, por exemplo, que a Fonte 1 que inicia a sua actividade mais cedo é a que leva mais tempo a ser drenada, o que traz uma implicação directa nos atrasos sofridos.

A tendência deste algoritmo faz com que os valores de X das várias fontes em competição sejam coincidentes (ver gráfico (b) da Figura 6.6), o que resulta numa distribuição da largura de banda equitativa (ver gráfico (e) da Figura 6.6), durante o tempo em que esses valores se mantiveram sobrepostos, só que, para conseguir essa situação,

as fontes que iniciam a sua actividade mais sedo são penalizadas em relação às que começam mais tarde.

6.1.3 Conclusões

A principal conclusão que se pode extrair desta primeira solução é a tendência para a distribuição equitativa da largura de banda pelos fluxos em competição (ver Figura 6.6), mas sem conseguir alcançar os restantes objectivos inicialmente propostos. Verificou-se em quase todos os resultados os mesmos problemas, nomeadamente a falta de equidade na distribuição dos atrasos porque o tempo de serviço concedido às fontes não era equitativo e, essencialmente, não tinha em atenção o quanto uma fonte era beneficiada em relação às outras.

6.2 Segunda solução

Nesta secção será apresentado uma segunda solução algorítmica, com base no método desenvolvido, de forma a resolver os problemas detectados na secção anterior, bem como uma solução intermédia (esta solução será apresentada apenas porque tem algumas características interessantes), que serviu de "ponte" entre a solução da secção anterior e a segunda solução.

6.2.1 Solução intermédia

Na sequência da solução precendente foi desenvolvida uma alternativa que se baseia na utilização quase exclusivamente dos valores de X para controlar os débitos atribuídos a cada fonte em competição. Esta solução baseia-se, essencialmente, no controlo dos valores máximo e mínimo de X e a fonte que está a transmitir é servida enquanto o seu valor de X não ultrapassar o valor máximo de X, obtido entre todos os valores de X de todas as fontes com células para servir. Assim que a fonte que está a ser servida tiver um valor de X igual ou superior ao valor de X máximo, então a fonte que passa a ser servida é a que apresenta um valor de X mínimo.

Esta solução é interessante do ponto de vista da complexidade porque é extremamente simples e tem em consideração a "história" passada das fontes, o que não acontecia nas soluções discutidas nos Capítulos 4 (ou nos resultados apresentados no Anexo C).

Tabela 6.1: Resultados numéricos que mostram as deficiências da primeira solução intermédia.

Parâmetros	Tempo (ms)													
	5	*7.5*	*10*	*12.5*	*15*	*16*	*17*	*18*	*20*	*21*	*23*	*24*	*25*	*26*
F_1	47	71	94	118	142	151	160	170	189	189	189	189	189	189
z_1	115	97	80	61	43	37	30	22	8	11	15	18	20	22
x_1	**35**	29	23	17	11	8	**6**	14	28	**34**	**30**	**35**	**34**	**32**
Q_1	0	24	47	72	96	105	114	114	114	105	105	96	96	96
F_2	1	24	47	71	94	104	113	123	142	151	170	179	189	189
z_2	149	132	115	97	80	72	65	58	43	37	22	16	8	11
x_2	1	18	11	**5**	22	30	**37**	35	31	28	**34**	**31**	**23**	**29**
Q_2	0	0	24	48	48	48	48	57	76	85	94	103	119	110
F_3		1	24	47	71	80	90	99	118	127	146	156	165	175
z_3		149	132	115	97	90	82	76	61	55	41	33	26	19
x_3		1	18	**35**	29	27	26	23	19	**16**	**30**	**29**	**36**	**34**
Q_3		0	0	0	24	33	42	51	70	79	79	88	88	97

Alguns resultados com interesse foram obtidos através da simulação, mas verificou-se que algumas configurações evidenciaram outros problemas que não tinham sido considerados inicialmente, o que se revelou útil para a apresentação de uma outra alternativa que será discutida nas secções seguintes.

6.2.2 Problemas

A detecção de inconsistências com a solução apresentada teve origem num teste numérico que incidiu sobre uma configuração com três fontes, em que o início de actividade das fontes foi decalado no tempo, mas com intervalos irregulares (0, 5, 7.5 ms). Assim que a simulação se iniciou, em especial com a entrada da terceira fonte em actividade, começaram a verificar-se alguns problemas relativamente aos valores máximos e mínimos de X, que eram os valores de referência para a transferência de controlo entre as fontes a serem escalonadas.

Para acompanhar esta discussão e mostrar em termos quantitativos onde se detectaram os problemas, recorrer-se-á à Tabela 6.1, apresentando um conjunto de valores relativos à simulação no período dos 0 aos 26 ms. Como se pode verificar a distribuição dos períodos de transmissão começa a ser desigual logo no início (por exemplo, a segunda fonte apenas transmite no início durante 2.5 ms, enquanto que a primeira e a terceira fonte transmitem durante 5 ms, como pode ser comprovado pelos valores de X dos instante de tempo 5, 7.5, 10 e 12.5 ms) e, com o decorrer do tempo, essa situação agrava-se e degenera no problema encontrado na solução da secção anterior, isto é, os valores de X vão encontrar-se (ver as colunas referentes aos instantes de tempo 23, 24,

25 e 26 ms); a partir desse momento, o controlo vai ser partilhado pelas fontes de forma equitativa, sem ter qualquer preocupação com as fontes mais ou menos beneficiadas até ao momento, isto é, o mesmo problema encontrado na primeira solução.

A ideia de que os valores máximos e mínimos de X eram suficientes para a resolução de parte dos problemas foi influenciada por algumas configurações utilizadas nos testes de simulação. Em situações em que a decalagem entre as fontes é igual, os valores de X obtidos são iguais para cada uma das fontes. Quando se alterou o valor dos intervalos verificou-se a situação apresentada na Tabela 6.1.

6.2.3 Solução algorítmica

Com base nos problemas apresentados na secção anterior procedeu-se ao desenvolvimento de uma nova estratégia para a sua resolução. A solução em discussão baseia-se na divisão do tempo em períodos fixos numa fase inicial e depois, durante a evolução do sistema, o valor do intervalo vai-se adaptando. Contudo, um dos aspectos importantes é a escolha do valor inicial do período de tempo, designado por k. Por conveniência foi escolhido um valor grande (dado que na realidade a decalagem entre as fontes é muito pequena), para que fosse possível verificar se a solução algorítmica encontrada conseguia alcançar os objectivos previamente estabelecidos. Inicialmente é possível tentar um conjunto de valores empíricos e, numa fase posterior, o valor ser escolhido de acordo com os valores que são transmitidos ao ED, durante o estabelecimento da conexão.

Os parâmetros derivados no modelo vão desempenhar um papel fundamental no desenvolvimento desta solução, por exemplo, o parâmetro Z servirá para controlar, na fase inicial, a entrada das fontes no sistema, enquanto que o valor do parâmetro X garante que nenhuma fonte transmite abaixo do seu débito médio garantido. É com base no valor de Z que o controlo do escalonador vai passar para a fonte que inicia o seu *burst*, porque é aquela que apresenta o valor de Z máximo (valor relativo, isto é, Z/b) e, desta forma, assim que as fontes iniciam a sua actividade, recebem o controlo se entretanto nenhuma das fontes em espera para serem servidas não apresentarem um valor de X negativo.

Após as fontes estarem em actividade o controlo passa a ser efectuado através da verificação dos intervalos de tempo. Para ilustrar a ideia pode-se observar a Figura 6.7 que mostra o tempo dividido em intervalos de comprimento fixo, k. Numa fase inicial

Figura 6.7: Intervalos fixos de tamanho k.

e para arranque do sistema deve-se escolher um valor base que depois se irá adaptando à medida que outras fontes forem iniciando a sua actividade. Com esta situação pode surgir a primeira questão: Se o valor escolhido para o intervalo de tempo for inferior ao instante de tempo em que a segunda fonte inicia a sua actividade o que acontecerá? Durante o tempo em que apenas estiver uma fonte em actividade o valor do intervalo quando alcançado é ignorado e o tamanho do intervalo considerado é o valor entre o início da primeira e da segunda fonte.

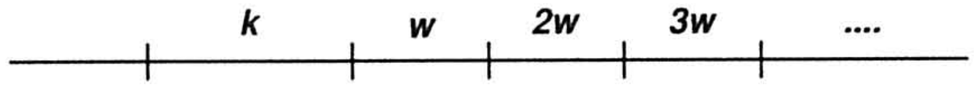

Figura 6.8: Intervalos de tamanho w com $w < k$.

No momento em que a segunda fonte inicia a sua actividade o tempo de intervalo mantém-se o mesmo, até ao valor de k, mas assim que uma terceira fonte entra em funcionamento antes de k ter sido alcançado, o novo valor do intervalo seguinte será w, isto é, o tempo entre o início da segunda e da terceira fonte. Após a alteração do valor de k, os intervalos subsequentes terão todos o mesmo valor, mas agora igual a w (ver Figura 6.8). Se existir a entrada de mais alguma fonte então o processo repete-se.

Como conclusão, o método baseia-se no controlo do valor máximo relativo de Z_i/b_i, que permite saber qual a fonte que está a iniciar a sua actividade no início do *burst* e pelo controlo dos intervalos de tempo, como pode ser observado no algoritmo seguinte.

/* Inicialização */

01. $if\ (N == 0)$
02. $\quad \forall_i(flag_i \leftarrow 0)$
03. $interval \leftarrow k$ **/* em** μs ***/**
04. $Fonte \leftarrow -1$
05. $F_{max} \leftarrow -1$

NQ	Número de filas com células
$FLAG$	Fonte que foi escalonada no *slot time* anterior
t	Tempo de espera da fonte para ser escalonada
T_{max}	Tempo máximo de espera para escalonamento
F_{max}	Fonte que está à mais tempo à espera para ser escalonada
$delay_i$	Última vez que a fonte i foi escalonada
$interval$	Intervalo de tempo definido à *priori*
$tInterval$	Contém o tempo de progressão dentro do período $interval$

06. $Max \leftarrow -999.0$
07. $t \leftarrow 0.0$
08. $timeInterval \leftarrow 0.0$
09. $T_{max} \leftarrow -999.0$
10. $cellSchedulingTime \leftarrow 424.0/DébitoDisponível$

/* Cálculo do valor de x e da fonte que está à mais tempo à espera para transmitir */

11. $do\ for\ i = 1\ to\ N$
12. $\quad if\ (Q_i > 0)$
13. $\quad\quad x_i \leftarrow b_i - (Z_i + Q_i)$
14. $\quad\quad t \leftarrow time() - delay_i$
15. $\quad\quad if\ (t >= T_{max})$
16. $\quad\quad\quad T_{max} \leftarrow t$
17. $\quad\quad\quad F_{max} \leftarrow i$
18. $\quad\quad NQ++$

/* Incremento do intervalo de tempo */

19. $if\ (NQ > 0)$
20. $\quad tInterval \leftarrow tInterval + cellSchedulingTime$

/* Cálculo da próxima fonte a ser escalonada */

21. $do\ for\ i = 1\ to\ N$
22. $\quad if\ (Q_i > 0)$
23. $\quad\quad if\ (x_i >= 0)$ **/* Valor de x positivo */**
24. $\quad\quad\quad if\ (Z_i/b_i > max\ and\ tInterval <= interval\ and\ flag_i == 0)$
25. $\quad\quad\quad\quad Max \leftarrow Z_i/b_i$
26. $\quad\quad\quad\quad Fonte \leftarrow i$
27. $\quad\quad\quad else$
28. $\quad\quad\quad\quad if\ (flag_i == 0\ and\ tInterval > interval)$
29. $\quad\quad\quad\quad\quad flag_i \leftarrow 1$
30. $\quad\quad\quad\quad\quad Fonte \leftarrow F_{max}$
31. $\quad\quad\quad\quad\quad flag_{F_{max}} \leftarrow 0$
32. $\quad\quad else$ **/* Situações em que o x é negativo */**
33. $\quad\quad\quad flag_i \leftarrow 0$
34. $\quad\quad\quad Fonte \leftarrow i$
35. $\quad\quad if\ (NQ == 1\ and\ tInterval > interval)$
36. $\quad\quad\quad tInterval \leftarrow 0.0$
37. $\quad\quad else$ **/* Verificar se outra fonte reuniu condições para ser escalonada */**
38. $\quad\quad\quad if\ (Fonte\ != FLAG\ and\ Fonte\ != -1)$

39. $flag_{FLAG} \leftarrow 1$
40. $FLAG \leftarrow Fonte$
41. $tInterval \leftarrow 0.0$
42. *else*
43. $flag_i \leftarrow 0$

/* Verificar se alguma condição falhou */

44. $if(Fonte == -1\ and\ NQ\ != 0)$
45. $Fonte \leftarrow F_{max}$
46. $flag_{F_{max}} \leftarrow 0$
47. $FLAG \leftarrow Fonte$
48. $tInterval \leftarrow 0.0$

/* Escalonar célula da fonte Fonte */

49. $if(Q_{Fonte} > 0)$
50. $delay_{Fonte} \leftarrow time()$
51. $Q_{ABR} \leftarrow Q_{Fonte}$

6.2.4 Resultados de simulação

A segunda solução foi testada através de simulação, com o objectivo de avaliar os méritos do escalonamento dos fluxos de tráfego, baseados em parâmetros do TB e no cálculo dos períodos de tempo de serviço atribuídos. Foram consideradas condições idênticas às do primeiro algoritmo.

Cenário 1

Numa primeira observação os resultados apresentados na Figura 6.9 permitem, desde logo, ficar com a sensação que os objectivos iniciais estavam atingidos, isto é, a ocupação dos *buffers* está controlada e a distribuição dos atrasos é equitativa. De facto, para este cenário os objectivos foram todos atingidos e para confirmar basta observar o gráfico (d) da mesma figura que apresenta uma distribuição dos atrasos relativamente correcta (a Fonte 1 sofre um atraso ligeiramente superior ao da Fonte 2) entre as duas fontes em consideração, mesmo com as fontes a começarem decaladas no tempo em 5 ms.

Da mesma forma que para os resultados apresentados para a primeira solução, os valores de X apresentados (ver gráfico (b) da Figura 6.9) para as duas fontes chegam para confirmar a forma como a solução se comporta perante este cenário. Como se pode observar no gráfico (b) estes valores são representados para ambas as fontes, onde se pode verificar que quando a Fonte 1 está a ser servida o seu valor de X está a crescer

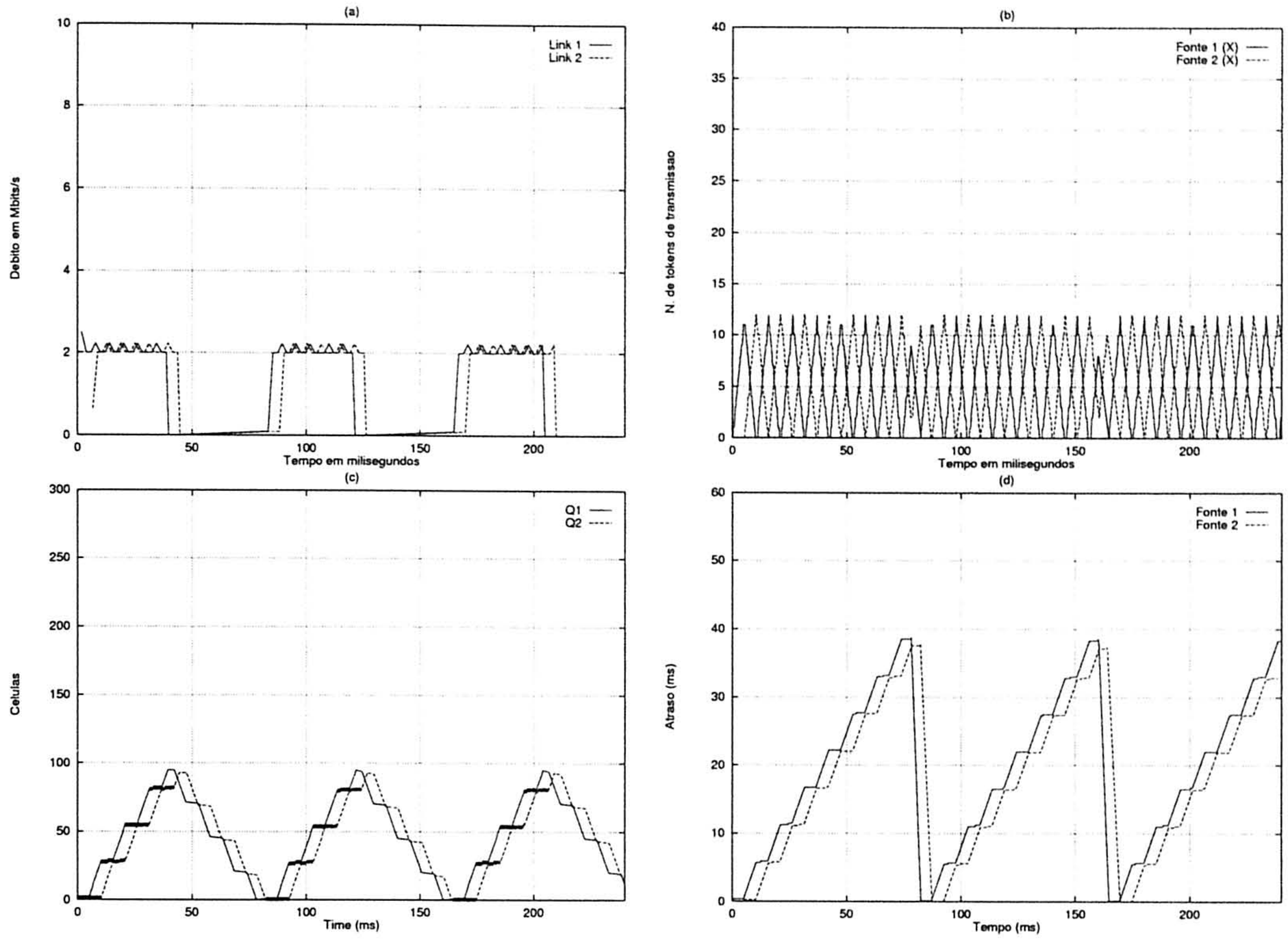

Figura 6.9: Resultados obtidos com o segundo algoritmo no Cenário 1.

linearmente enquanto o valor de X da segunda está a decrescer. É necessário referir que o problema detectado na primeira solução (os valores de X tinham a tendência de se manter "juntos" quando se encontravam) é ultrapassado, porque quando os valores de X se cruzam nada acontece e cada um continua a sua progressão ascendente ou descendente, respectivamente.

Cenário 2

Como já referido, o objectivo desta simulação era verificar o comportamento do algoritmo numa configuração um pouco peculiar porque tenta-se realizar uma distribuição equitativa dos atrasos, quando uma fonte praticamente terminou a sua transmissão e uma segunda está apenas no seu início. Os resultados da Figura 6.10 mostram que esse objectivo foi relativamente conseguido, uma vez que os atrasos médios obtidos por cada cada uma das fontes foi sensivelmente de 5 ms, quando o tempo de decalagem entre elas é de 37.4 ms.

Numa perspectiva simplista (sem verificar os resultados dos restantes cenários) poder-se-ia dizer que o comportamento dos valores de X são os ideais, uma vez que a

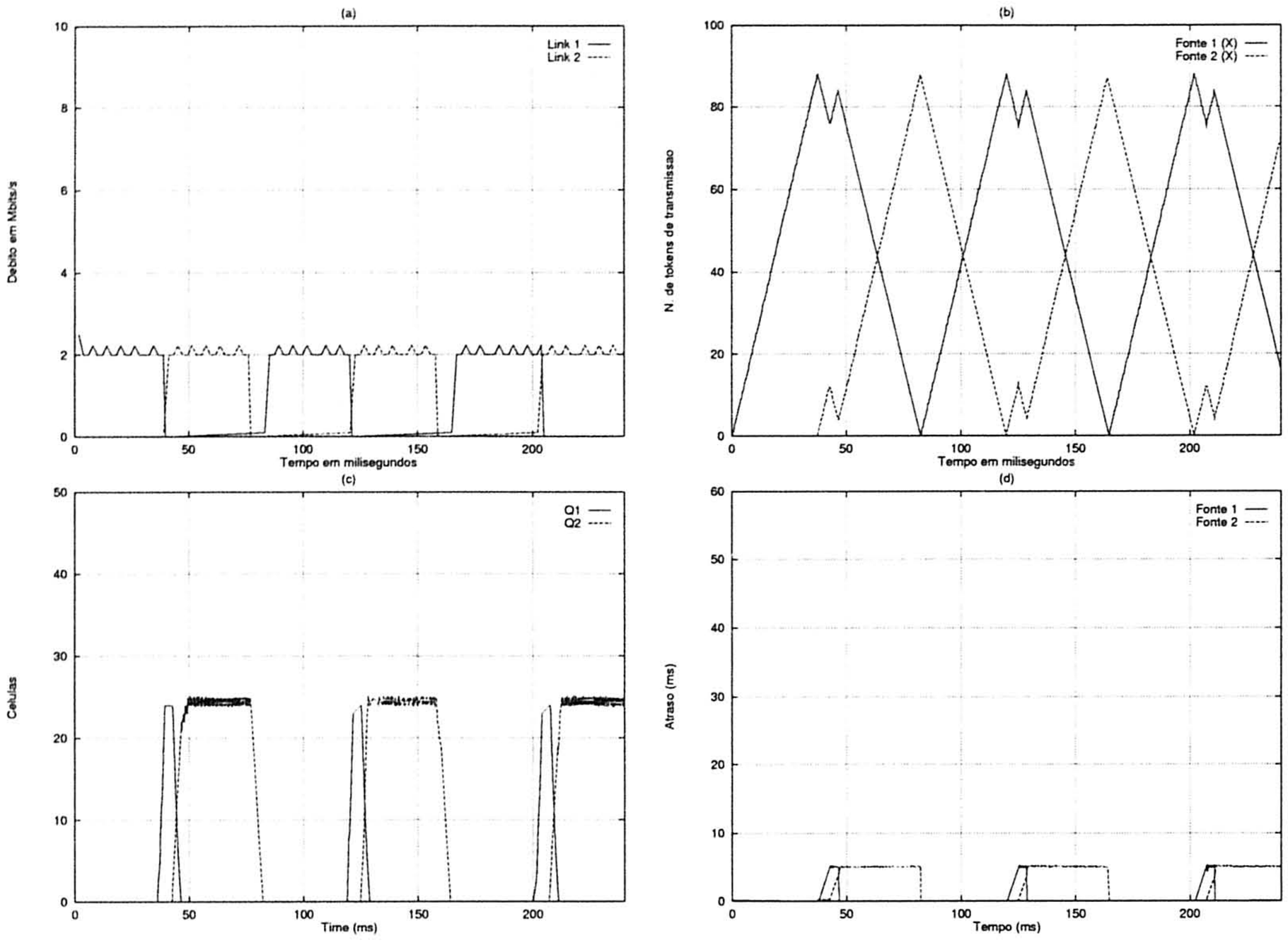

Figura 6.10: Resultados obtidos com o segundo algoritmo no Cenário 2.

troca do controlo é efectuado e os valores de ambas as fontes estão bem afastados (como pode ser observado nos gráfico (b) da Figura 6.10). Esta situação era um dos problemas que afectava a solução dos valores máximo e mínimo de X, mas com a introdução do controlo temporal é possível resolver esta questão sem grande complexidade adicional.

Cenário 3

As vantagens desta solução com este tipo de cenário são visíveis através dos resultados apresentados na Figura 6.11, porque através da observação dos gráficos (c) e (d) verifica-se que, tanto a ocupação dos *buffers* como a distribuição dos atrasos são os esperados. A utilização dos resultados apresentados no gráfico (b) da mesma figura servem de forma complementar para análise e, como pode ser observado, os valores de X mostram o comportamento do escalonador na forma como o serviço é distribuído pelas fontes em competição.

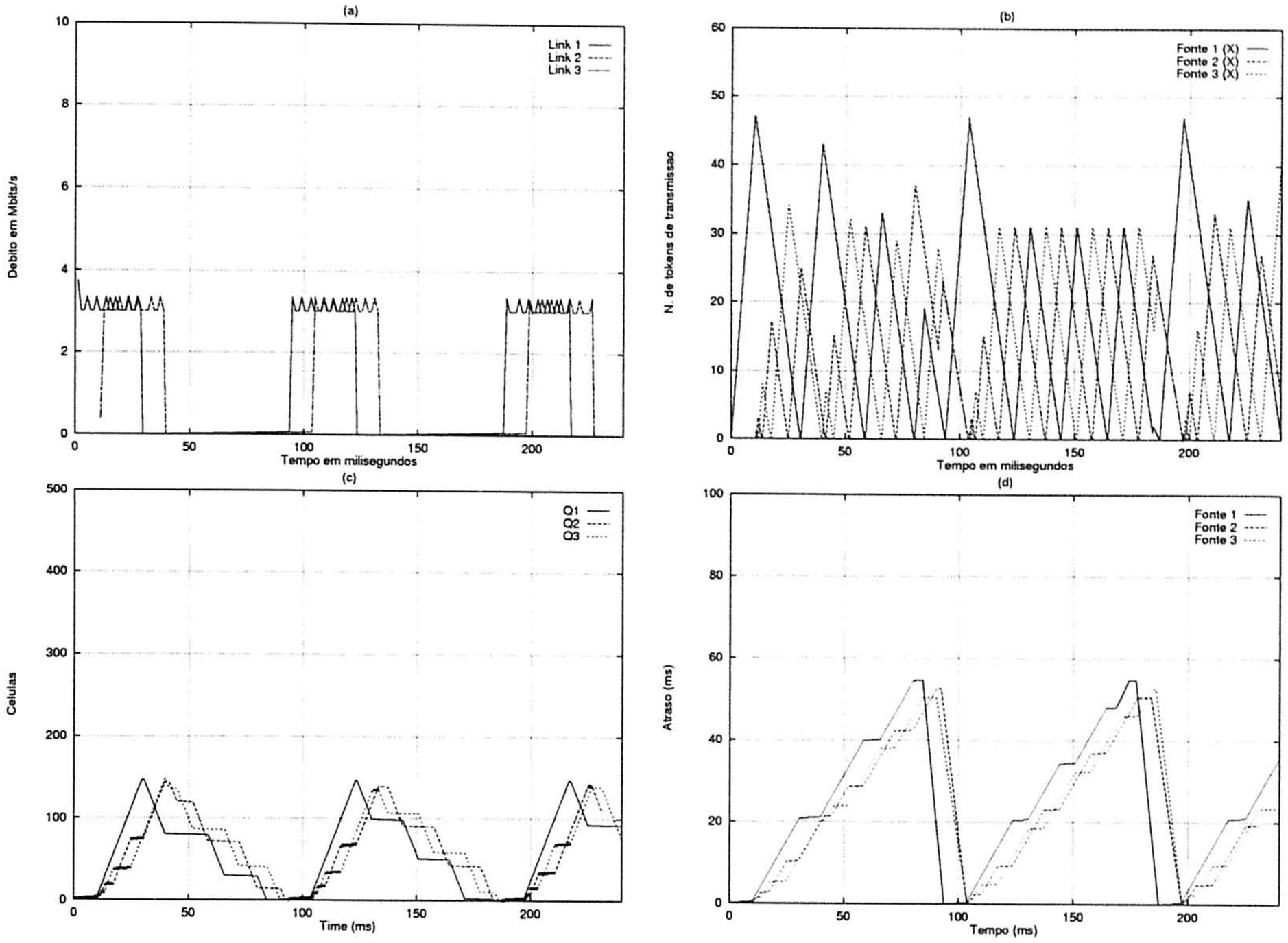

Figura 6.11: Resultados obtidos com o segundo algoritmo no Cenário 3.

Cenário 4

A distribuição da largura de banda e a respectiva ocupação dos *buffers* a jusante dos TBs podem ser observadas na Figura 6.12, por exemplo, o gráfico (c) apresenta uma ocupação com as devidas proporções relativamente aos débitos de geração de tráfego de cada uma das fontes, isto é, a ocupação do *buffer* da segunda fonte é três vezes superior ao da primeira; no entanto, relativamente às distribuições dos atrasos (gráfico (d)), não apresenta essa proporção mas uma distribuição similar.

Os resultados do gráfico (b) complementam a discussão anterior, verificando-se a influência do teste do valor de X ser ou não inferior a zero, uma vez que a Fonte 2 transmite durante 5 ms e a Fonte 1 transmite até que o valor de X da Fonte 2 seja igual a zero, até que o serviço volta a ser atribuído à Fonte 2. Desta forma a Fonte 2 é servida mais vezes do que a Fonte 1 e o atraso é distribuído equitativamente.

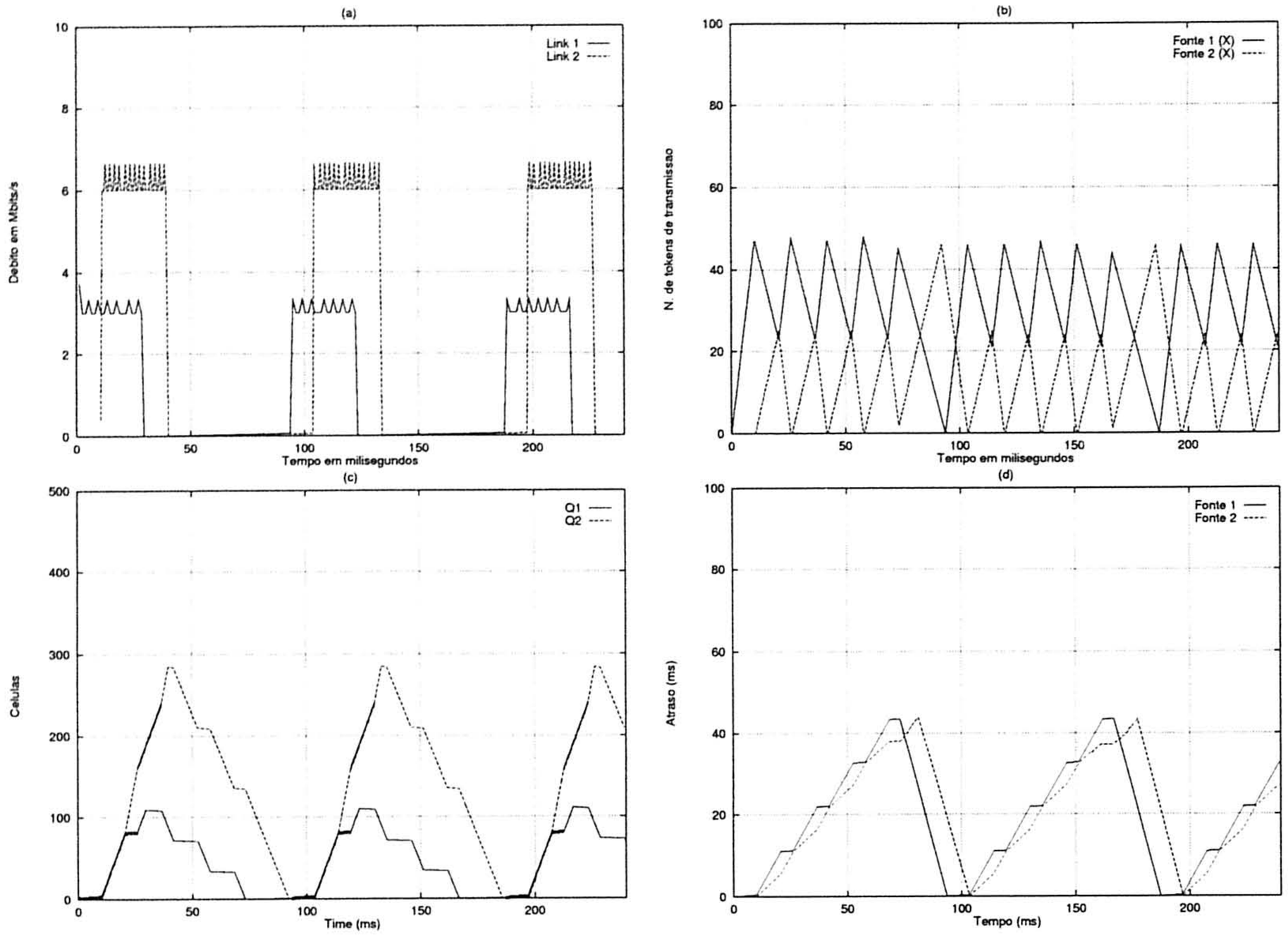

Figura 6.12: Resultados obtidos com o segundo algoritmo no Cenário 4.

Cenário 5

Com esta solução, aumentando o grau de complexidade do cenário é possível obter resultados interessantes (ver Figura 6.13), por exemplo, a ocupação dos recursos é total e a distribuição da largura de banda é realizada de forma equitativa, mesmo quando as fontes iniciam a sua actividade em diferentes períodos de tempo. A observação dos gráficos (c) e (d) da mesma figura permite perceber a forma como a distribuição da largura de banda é realizada sem beneficiar nenhuma das fontes, independentemente do seu início de actividade. O padrão encontrado na ocupação dos *buffers* a jusante dos TBs e da consequente distribuição dos atrasos prova isso mesmo.

Da mesma forma que nas discussões dos resultados dos cenários anteriores, a avaliação dos resultados apresentados no gráfico (b) da Figura 6.13 serve para consubstanciar a análise já efectuada. Dos resultados apresentados é possível destacar os valores de X de cada uma das fontes, para verificar a forma como a distribuição do serviço do escalonador foi realizado. Verifica-se que em qualquer dos casos este valor tem sempre uma variação entre 0 e 35, para além de nunca ser inferior a zero, por exem-

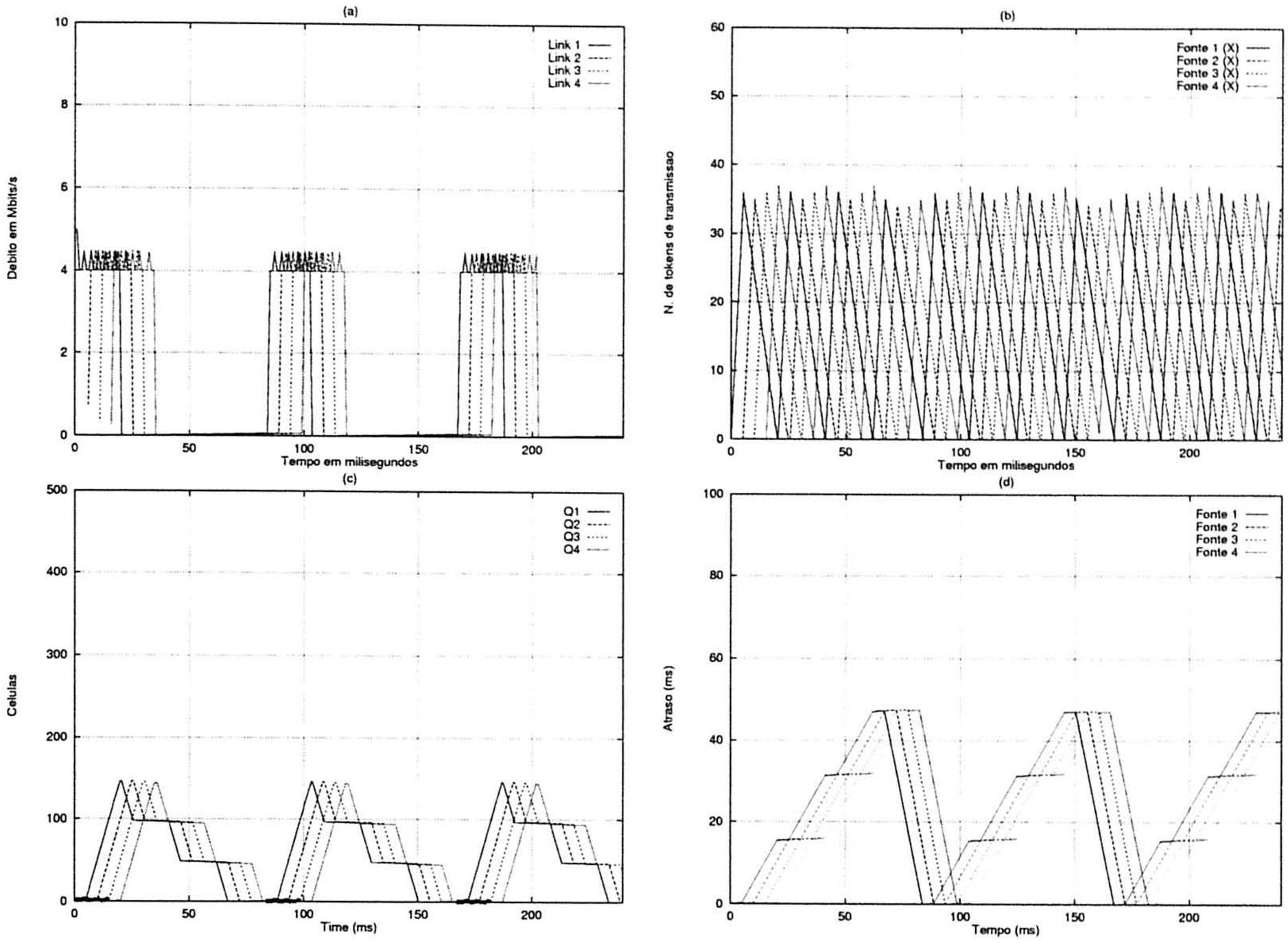

Figura 6.13: Resultados obtidos com o segundo algoritmo no Cenário 5.

plo, no gráfico (b) da mesma figura são apresentados os valores de X sobrepostos onde se pode verificar que no instante em que um dos valores de X está crescer, os valores de X das outras fontes estão a diminuir.

Estes resultados provam, a par dos resultados obtidos com os outros cenários, a forma simples e eficiente como o algoritmo, com base nos valores dos parâmetros dos TBs e dos intervalos de tempo dinamicamente calculados, podem fazer uma distribuição equitativa da largura de banda e, consequentemente, ter em atenção quais as fontes já beneficiadas inicialmente e a forma como as outras devem ser compensadas, sem prejudicar as primeiras.

Cenário 6

Para confirmar o desempenho do algoritmo relativamante a este cenário basta uma simples observação da Figura 6.14, nomeadamente dos gráfcos da ocupação dos *buffers* (c) e da distribuição dos atrasos (d), para se concluir que o algoritmo apresenta um bom desempenho e cumpre os objectivos inicialmente propostos.

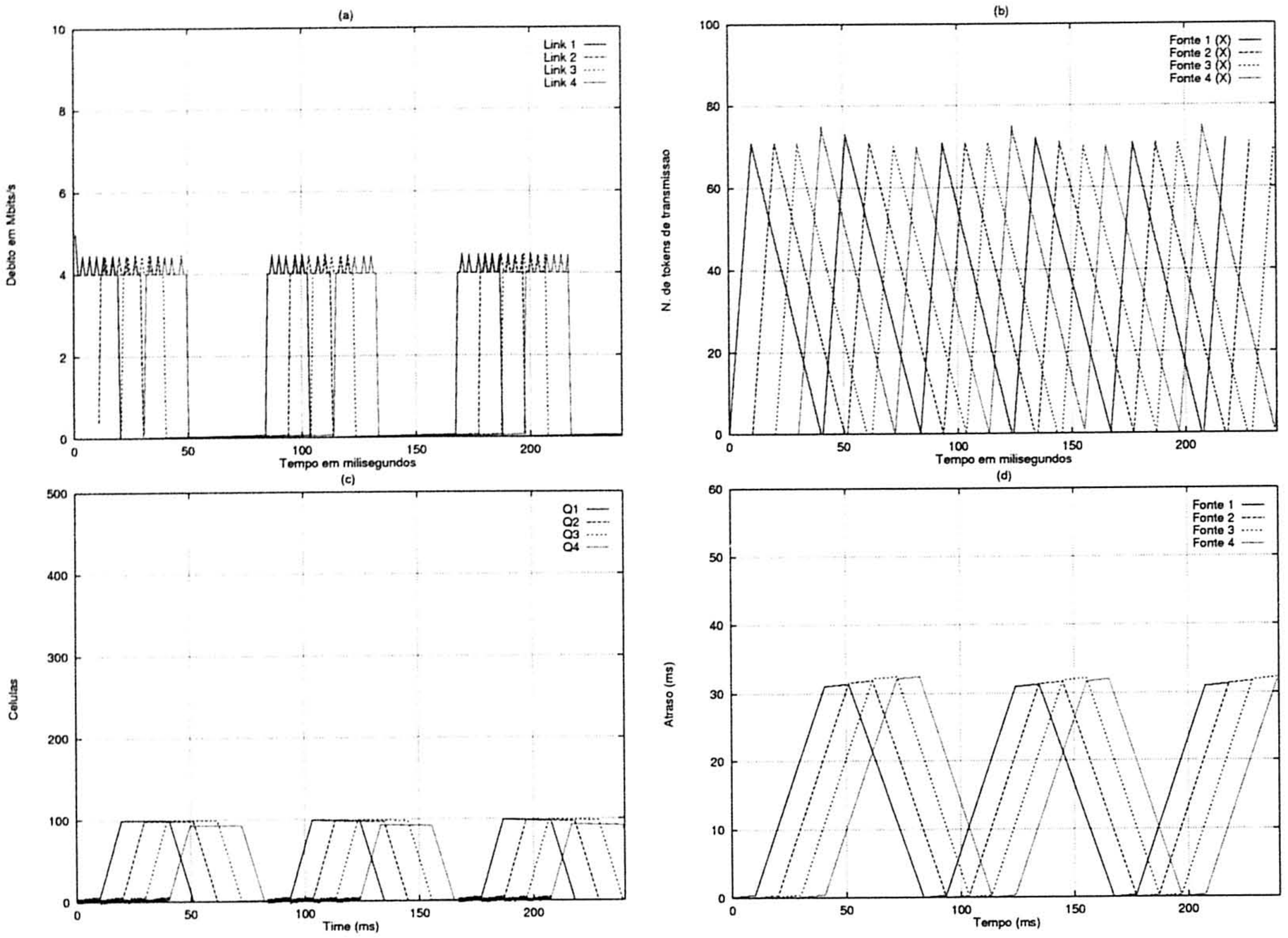

Figura 6.14: Resultados obtidos com o segundo algoritmo no Cenário 6.

6.2.5 Conclusões

Após a detecção dos problemas das soluções anteriores procedeu-se ao estudo de mais uma solução que fosse capaz de ultrapassar essas deficiências. A ideia subjacente ao modelo encontrado foi dividir o tempo em períodos pré-definidos, um pouco à imagem dos métodos de escalonamento da classe *non-Workconserving*, do tipo *slotted.*

Os resultados obtidos apontaram para um modelo interessante que apresenta uma "correcta" distribuição da largura de banda, da ocupação das filas de espera, dos atrasos médios e fundamentalmente na relativa simplicidade de implementação. Contudo, existe um problema que necessita de mais reflexão e estudo que é como definir o intervalo inicial porque, por exemplo, se o valor for demasiado curto pode provocar demasiadas passagens de controlo.

Como será verificado nas soluções seguintes, o recurso a intervalos de tempo deixou de ser utilizado, uma vez que o cálculo inicial do intervalo e os sucessivos ajustamentos não se mostraram os ideais. Por isso, nas propostas seguintes, o tipo de controlo utilizado passou a ser discreto, o que levou a uma solução interessante, relativamente à

forma como o serviço é atribuído a cada fluxo, bem como no que se refere à complexidade de implementação.

6.3 Terceira solução

Nesta secção será apresentada uma segunda solução intermédia que utiliza o conceito de intervalos de tempo calculados dinamicamente, e que serviu de base para a terceira e última solução.

6.3.1 Solução intermédia

Com base no conjunto de problemas da proposta anterior, em que se pode destacar o cálculo do valor do intervalo de tempo e, essencialmente, a falta de um valor de referência para que seja possível o controlo na passagem de serviço entre fontes, definiu-se uma nova solução. Como se tem observado até ao momento, existem duas situações limites (ver Figura 5.4 e 5.5) que devem ser as primeiras a ser consideradas.

A primeira situação, apresentada na Figura 5.4, implica apenas uma mudança de serviço entre as duas fontes em competição, por intervalos de tempo com o mesmo valor (o valor é a diferença entre o início de actividade da primeira e da segunda fonte). Qualquer solução que atribui serviço de escalonamento durante intervalos de tempo com dimensão fixa, garante uma distribuição justa da ocupação dos *buffers* e da distribuição dos atrasos. Contudo, se a situação em discussão for a inversa, isto é, uma fonte que inicia a sua actividade no início da simulação e uma segunda fonte inicia a sua actividade quase no final do *burst* da primeira, a solução apresentada para o primeiro caso já não se aplica.

Como pode ser observado na Figura 5.5, o instante de tempo em que a segunda fonte inicia a sua actividade já ultrapassou metade do *burst* da primeira. É precisamente esta situação que vai causar parte dos problemas quando se pretende ter uma única solução para resolver as duas. Com base nos problemas encontrados e na "experiência" obtida com as soluções já apresentadas verificou-se que existe um padrão em todas elas, isto é, o intervalo de tempo entre o início da actividade da primeira e da segunda fonte apresenta-se como o principal problema a resolver.

A partir do instante em que se percebeu que para a situação apresentada an-

teriormente é possível encontrar uma solução, isto é, usar simplesmente para a primeira situação (ver Figura 5.4) o intervalo de tempo entre a primeira e a segunda fonte como o intervalo de medida do tempo de serviço que deve ser atribuído a cada uma das fontes, enquanto que para a segunda situação (ver Figura 5.5) deve ser usado o intervalo de tempo que ainda falta para terminar a transmissão da primeira.

A implementação de uma solução deste tipo pode parecer simples, mas de facto não é bem assim, porque como será observado, é necessário saber quando é que um *burst* começa e acaba, que tipo de informação deve ser guardada, etc. Inicialmente é necessário guardar mais informação por cada uma das portas de entrada, do que nas soluções anteriores. O instante de tempo em que o *burst* se inicia e o tempo que o *burst* vai demorar são informações cruciais para a implementação desta solução.

A obtenção do início do *burst* é conseguida apenas guardando o tempo de chegada da primeira célula, enquanto que para saber o tempo de duração do *burst* se utilizou a informação que é indicada ao ED aquando da realização do controlo de admissão, uma vez que é necessário saber o número máximo de células do *burst* e o débito à qual elas são geradas. Contudo, é necessário indicar que esta operação deve ser realizada apenas para a primeira célula do *burst* (considerando os cenários definidos na secção 5.5), e para executar esta tarefa é preciso utilizar uma variável de estado que indica se a primeira célula já foi processada. Assim que a primeira célula do *burst* chega, a variável de estado é atribuída a 1 e só volta a 0 quando o *burst* acabar.

Com a marcação do início do *burst* e a obtenção do seu tamanho já é possível implementar o método descrito; no entanto, ainda está por resolver o problema de saber se o instante de tempo em que a segunda fonte inicia a sua actividade já ultrapassou mais de metade do *burst* da primeira. Antes de resolver esta situação é necessário indicar de que forma é que se identifica se uma fonte iniciou ou não a sua actividade. A utilização de duas variáveis ($Fonte$, $Fonte^{Old}$) é suficiente para realizar este propósito, uma vez que a variável $Fonte$ vai conter a fonte seleccionada para ser escalonada no *slot time* (tempo de célula) corrente, enquanto que a variável $Fonte^{Old}$ contém a fonte que foi escalonada no *slot time* anterior. Uma simples comparação entre as duas variáveis permite verificar duas situações: a primeira serve para confirmar se uma segunda fonte iniciou a sua actividade e a segunda para verificar se existiu uma troca de controlo entre fontes já a transmitir.

A partir do momento em que se verifica que o valor das variáveis é diferente é necessário calcular o próximo intervalo de tempo que será utilizado nos tempos de serviço atribuídos a cada fonte. O primeiro problema a resolver é verificar se a segunda fonte, ao iniciar a sua actividade, ultrapassou mais de metade do *burst* da primeira. Considerando que a primeira fonte já está a transmitir e uma segunda inicia a sua actividade, admite-se que $Fonte^{Old}$ representa a primeira fonte e $Fonte$ a segunda. Com base no tempo corrente e no tempo de início do *burst* da primeira fonte é possível saber o tempo de transmissão já efectuado e verificar se a metade já foi ultrapassado. Neste instante é possível questionar esta solução porque desta forma vão ser contemplados *bursts* que estejam em transmissão e que a dada altura atinjam a metade, ou então a partir do momento que foi atingida a metade de um não se considere a metade do outro. Para a resolução deste problema foi necessário recorrer a mais uma variável de estado. O modo de funcionamento implementado é simples e evita problemas, porque assim que exista uma situação de mudança (o início de uma segunda fonte, por exemplo) este teste só deve ser realizado uma única vez, para verificar se o intervalo inicial foi ou não superior a metade do *burst*.

Assim que o teste é verificado e se confirma que foi ultrapassado a metade do *burst* da primeira então é considerado apenas o tempo complementar, como o intervalo de tempo a atribuir ao tempo de serviço que será disponibilizado pelo escalonador, a partir desse instante. Após o teste inicial é considerado sempre o intervalo com o tempo mínimo, como o intervalo de tempo que cada uma das fontes vai ter disponível para encaminhar as suas células.

As alterações introduzidas no algoritmo não parecem ser, à primeira vista, significativas quando se compara com os algoritmos que se têm vindo a estudar, uma vez que se introduziu apenas algumas variáveis de estado. Contudo, estas variáveis de estado permitirão um cálculo dinâmico do intervalo de tempo a atribuir a cada uma das fontes que era uma das dificuldades da versão anterior.

6.3.2 Problemas

A avaliação dos resultados da segunda solução intermédia ilustram uma melhoria significativa quanto ao cálculo do intervalo de tempo atribuído pelo escalonador a cada uma das fontes, relativamente às soluções precedentes. Contudo, a obtenção do intervalo

obrigava ao registo dos instantes de tempo em que o *burst* tinha início (tarefa que nos casos reais se torna muito complicada), e por consequência, implicava a utilização de uma variável de estado, para garantir que o instante de tempo era só registado apenas uma única vez em cada *burst*. A par desta situação era ainda necessário, no momento em que deixa de haver células para escalonar, colocar a variável de estado e de início do *burst* a zero. Estas situações motivaram o desenvolvimento de uma nova solução, uma vez que o tráfego gerado pelas fontes, apesar de ser do tipo *On-Off* (na realidade os períodos *On* e *Off* não são completamente definidos) é realizado com débitos reduzidos umas vezes e ao débito de pico noutras.

Como na solução anterior, um dos problemas que tem levantado algumas questões é a identificação do início e do fim do *burst*. Outra das questões é, por exemplo, no instante em que nenhuma fila apresenta células poder-se-ia pensar que todas as células já chegaram e já foram servidas; no entanto, se algum problema ocorreu (problemas com a fonte, por exemplo) é necessário arranjar uma forma para solucionar esta questão.

Uma outra situação que não tinha ainda sido contemplada pelas soluções anteriores, era o facto de existir uma fonte a transmistir com um débito inferior ao débito de serviço ($R_i^{in} < R^{out}$), o que vai resultar em instantes em que o número de filas com células (NQ) seja nulo o que obrigará, segundo a solução anterior, a uma reinicialização das variáveis, o que não é correcto porque podem existir ainda *bursts* que não foram completamente transmitidos.

6.3.3 Solução algorítmica

Com o mesmo propósito das soluções precedentes, o método é inicialmente estudado e aplicado a duas situações limite (ver Figura 5.4 e 5.5) que devem ser as primeiras a serem consideradas quando se pensa em desenvolver soluções de escalonamento que tenham os mesmos objectivos definidos inicialmente.

A primeira situação (apresentada na Figura 6.15 (a)) implica apenas uma mudança de serviço entre as duas fontes em competição, por intervalos de tempo com valor C_1. Contudo, se a situação encontrada for a representada na Figura 6.15 (b) já é necessário realizar alterações. Qualquer solução que atribui serviço de escalonamento durante intervalos de tempo com dimensão fixa garante o controlo da ocupação dos *buffers* e da distribuição dos atrasos. No entanto, se a situação em discussão for a

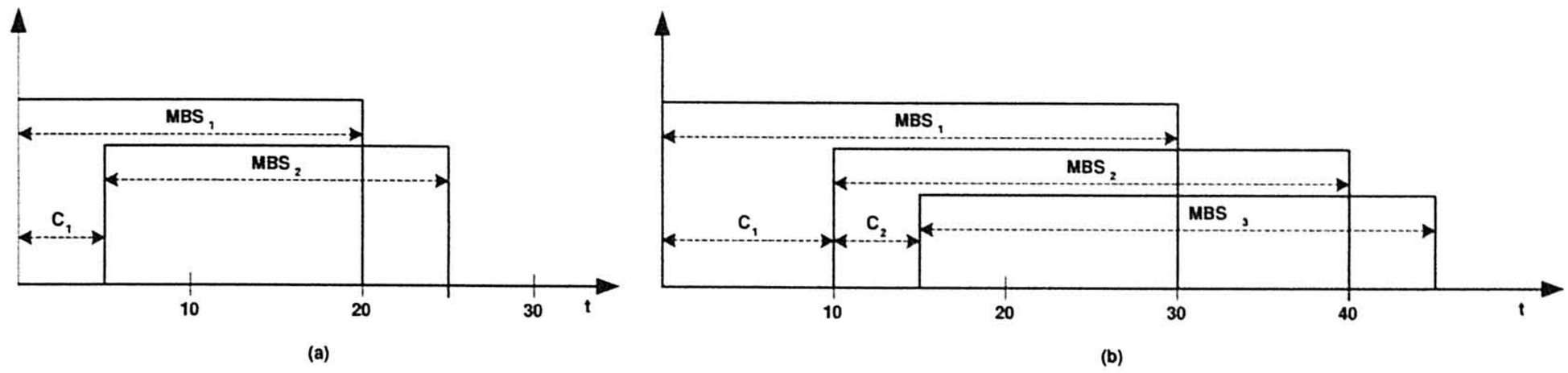

Figura 6.15: Situações de resolução mais simples.

inversa, isto é, uma fonte que inicia a sua actividade no início da simulação e uma segunda fonte que inicia a sua transmissão quase no final do *burst* da primeira, a solução apresentada para o primeiro caso já não se aplica.

Apesar de se estar a usar expressões como intervalos de tempo, a solução em estudo usa uma noção diferente, que é necessário introduzir antes de explicar o modo de funcionamento. Nesta solução será usada a noção de **créditos** que são atribuídos a cada uma das fontes pelo escalonador, isto é, o número de células consecutivas que vão ser escalonadas relativamente à mesma fonte. Esta ideia evita o cálculo do intervalo de tempo e não "obriga" ao conhecimento *preciso* do início e do fim do *burst*, porque o início é dado pelo número nulo de créditos consumidos por um fluxo e o fim pelo número máximo de células que o *burst* pode ter, dado por MBS, evitando problemas pelo facto da fonte não transmitir em períodos On e Off perfeitamente definidos.

A simulação da noção de créditos é realizada através da utilização de um contador para cada fonte que é incrementado por cada célula que é escalonada, o que vai permitir, no instante em que uma segunda fonte inicia a sua actividade, saber o número de créditos consumidos. A Figura 6.15 (a) ilustra o sistema mais simples e, como pode ser observado, no instante em que a segunda fonte inicia a sua actividade, a primeira já beneficiou de C_1 créditos. A partir deste instante o número de créditos atribuídos a cada uma das fontes é C_1. Uma situação como a da Figura 6.15 (b) é diferente porque o número de créditos consumidos entre o início da primeira e da segunda fonte é maior do que o número de créditos consumidos entre o início da segunda e da terceira fonte, logo o algoritmo adapta-se e passa a usar C_2 como valor de referência.

Na situação oposta às primeiras (Figura 6.16 (a) e (b)), o instante de tempo em que a segunda fonte inicia a sua actividade já ultrapassou metade do *burst* da primeira. Com base na solução encontrada para a situação anterior tomou-se a decisão de seguir

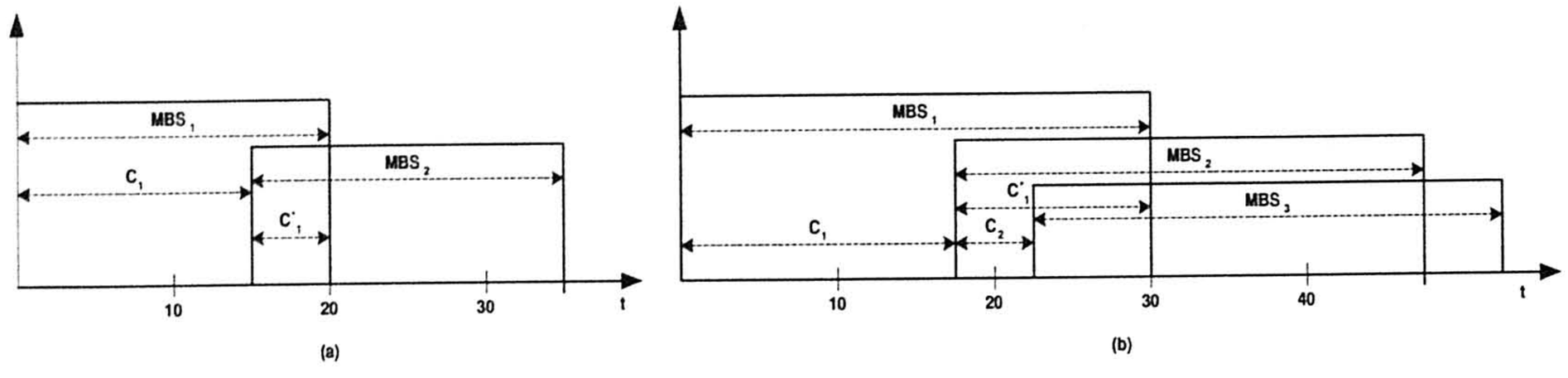

Figura 6.16: Situações com resolução contrária.

o mesmo princípio, só que para esta solução não é aplicada a noção de créditos usados, mas os que ainda faltam usar para poder transmitir todo o *burst*. É necessário explorar a ideia de *metade* do *burst* porque é este valor que vai permitir realizar o controlo em relação ao início de actividade de uma segunda fonte antes, ou depois do meio do *burst*, uma vez que dependendo do resultado, o tipo de operação será diferente. A ideia é simplesmente usar para a segunda situação (ver Figura 6.16 (a)) o número de células que ainda faltam ser escalonadas até terminar o *burst*, isto é, com base no valor do MBS_i e no número de crétidos já consumidos sabe-se quantos ainda faltam para servir o *burst* completo ($C_1^* = MBS_1 - C_1$).

A ocorrência de uma situação igual à representada na Figura 6.16 (b) obriga a uma nova adaptação do algoritmo na escolha do número de créditos de referência, isto porque, apesar do valor de referência ser o número de créditos que ainda estão em falta (C_1^*), este deve ser reajustado porque entretanto uma nova fonte iniciou a sua actividade e o número de créditos utilizado é inferior ao referenciado ($C_2 < C_1^*$), o que implica um novo reajustamento, sendo C_2 o novo valor de referência.

Tal como na solução anterior foi utilizado o tempo de transmissão da primeira fonte e o tempo de duração do seu *burst*; mais uma vez observa-se que é necessário saber o início e o fim do *burst*. Para resolver esta situação foram utilizados os parâmetros Z_i e b_i de cada uma das fontes que têm células para servir. Estes parâmetros vão permitir em conjunto saber se um determinado *burst* já recebeu créditos suficientes para servir mais de metade das suas células, através da expressão (5.22).

Sabendo que o método apresenta como valores iniciais $Z_i = b_i$ (*bucket* inicialmente cheio) e como o valor de Z_i vai diminuindo à medida que as células vão chegando ao ED e o valor de b_i se mantém constante, esta situação permite verificar se $\nu < 0.5$ já foi atingido. Nessa altura é possível saber com precisão o número de créditos ainda

necessários para servir o resto do *burst*.

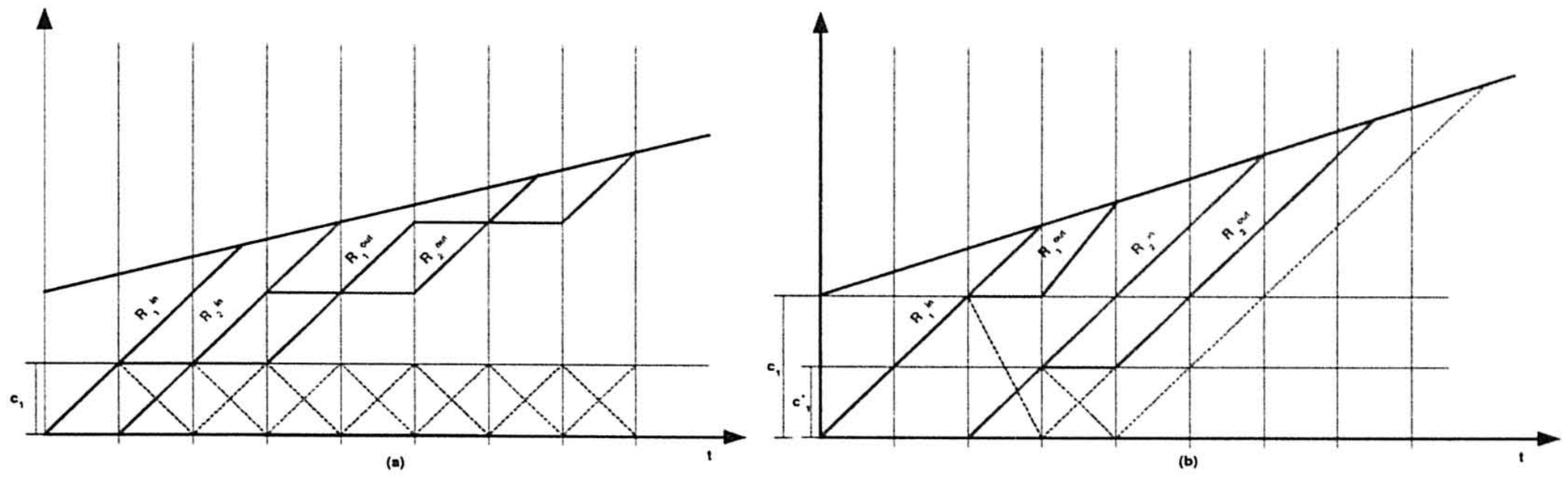

Figura 6.17: Exemplo da utilização dos créditos em ambas as situações.

A utilização da técnica de créditos é ilustrada na Figura 6.17 para as duas situações. Na situação mais simples (Figura 6.17 (a)) pode-se observar que a partir do instante em que a segunda fonte inicia a sua actividade é utilizado o valor C_1 como o número de créditos concedido em cada período a cada uma das fontes. Quando a situação é a contrária (Figura 6.17 (b)) é utilizado o número de créditos que ainda faltam servir (C_1^*) como referência relativamente ao que cada fonte pode consumir, a partir do instante em que a segunda fonte iniciou a sua transmissão.

A questão do início e do fim do *burst* pode provocar algumas dificuldades na obtenção de soluções de escalonamento, porque o facto das células do *burst* terem já chegado ao ED não implica que já tenham sido servidas. Com base na questão anterior é necessário arranjar uma forma de reinicializar os valores dos créditos para a recepção de um novo *burst*. No instante em que se verifica que nenhuma fila apresenta células poder-se-ia pensar que todas as células já chegaram e já foram servidas. A resolução desta questão é conseguida à custa de uma variável adicional que terá o número total de créditos consumidos e, assim que forem iguais ao número máximo de células do *burst* (MBS_i), confirma-se que o *burst* já chegou e já foi servido por completo. Se todas as fontes que estão com conexões activas se encontram nesta situação então um novo ciclo se avizinha e as variáveis são reinicializadas. Contudo, existe ainda a necessidade de verificar se uma das três situações ocorreu:

(i) se não existe nenhuma fonte para servir;

(ii) se existe uma fonte a transmitir com um débito inferior ao débito de serviço e;

(iii) se existe apenas uma fonte com células para serem servidas.

Na primeira situação basta reinicializar todas as variáveis para permitir um novo período. No segundo caso é necessário perceber que existe a possibilidade de uma fonte transmitir com um débito inferior ao débito de serviço. No instante em que o número de fontes a servir é supostamente nulo, é obrigatório fazer esta distinção, para não reinicializar os valores das variáveis que contêm o número de créditos de referência, o número total de créditos consumidos e o número de créditos consumidos para uma determinada fonte, evitando assim "viciar" o funcionamento do mecanismo. A terceira situação ocorre quando apenas existe uma única fonte com células para serem servidas, o que implica a manutenção da variável de estado de mudança em zero, para não confundir o algoritmo no momento em que um novo *burst* se inicia.

Outra das situações a ter em consideração é a necessidade de garantir que uma fonte nunca seja servida abaixo do seu débito médio garantido (SCR). O valor do parâmetro X, do método proposto, serve como elemento fundamental para o referido controlo, uma vez que sempre que o seu valor seja positivo ou nulo está garantido o serviço negociado; no entanto, sempre que o valor seja negativo é necessário passar de imediato o controlo de serviço para a fonte em questão. Contudo, a passagem de controlo de serviço não pode ser efectuada de qualquer forma, isto é, existe a necessidade de garantir que ficam guardados os créditos mínimos em uso como referência para a fonte que estava a ser escalonada, para que a próxima a ser servida possa beneficiar da mesma referência.

Para uma melhor ilustração desta ideia pode-se usar os resultados obtidos com o Cenário 4 (ver gráficos da Figura 6.21). Por exemplo, se uma fonte está a transmitir com o seu PCR e SCR que são o dobro do PCR e SCR de outra fonte, então o escalonador deve servir duas vezes mais a primeira do que a segunda. Se o número de créditos concedidos e consumidos pela primeira fonte for, por exemplo, quarenta e sete (número de células servidas até esse momento), a partir do instante em que a segunda fonte inicia a sua actividade (a fonte com um PCR e SCR que são o dobro do PCR e SCR da outra fonte) vai fazer com que o valor de X da primeira seja negativo, antes do número de créditos consumidos ser atingido, obrigando a trocar a fonte que estava a ser servida; no entanto, é necessário guardar o número de créditos consumidos incialmente, para que a primeira fonte também possa beneficiar e, a partir da primeira troca a segunda fonte

beneficia de quarenta e sete créditos enquanto que a primeira apenas de vinte e três.

6.3.4 Detalhes de implementação

A implementação desta solução é simples, mas necessita de recorrer a mais do que uma variável de estado. Ao contrário da solução anterior o problema de início e de fim do *burst* deixa de se colocar, uma vez que o início é dado pelo valor nulo dos créditos consumidos pelo fluxo e o fim é obtido simplesmente por comparação entre os créditos acumulados e o seu MBS. Um dos problemas que se depara neste tipo de solução é saber se no instante de tempo em que a segunda fonte inicia a sua actividade já foram consumidos mais créditos do que metade do número de células do *burst* da primeira. Mas de que forma é que se identifica se uma fonte iniciou ou não a sua actividade? Do mesmo modo que na solução precedente, utilizaram-se as variávies $Fonte$ e $Fonte^{Old}$ e através de uma simples comparação entre as duas é possível verificar se uma segunda fonte iniciou a sua actividade, ou se existiu uma troca de controlo entre fontes a transmitir.

A partir do instante em que se verifica que o valor das variáveis é diferente é necessário saber se o número de créditos utilizado é inferior ao valor inicialmente atribuído por omissão; na primeira fase é sempre inferior, o que vai fazer com que o primeiro valor real seja o número de créditos consumidos entre a primeira e a segunda fonte. Por outro lado, quando se considera que a primeira fonte já está a transmitir e uma segunda inicia a sua actividade, admitindo que $Fonte^{Old}$ representa a primeira fonte e $Fonte$ a segunda, e com base no valor de Z e no valor de b da primeira fonte ($Z_{Fonte^{Old}}$ e $b_{Fonte^{Old}}$), é possível verificar se a metade já foi ultrapassada ($if\ (Z_{Fonte^{Old}}/b_{Fonte^{Old}} < 0.5)$). Da mesma forma que na solução anterior é usada uma variável de estado ($tFlag$) para evitar problemas, porque assim que exista uma situação de mudança, este teste só deve ser realizado uma única vez, para verificar se o número de créditos consumidos inicialmente foi ou não superior a metade do número máximo de células do *burst*. Se o resultado é positivo então é considerado apenas o número de células ainda em falta, se for inferior ao número mínimo de créditos obtidos até esse instante.

As alterações introduzidas no algoritmo, nomeadamente, a utilização de variáveis de estado vai permitir um cálculo dinâmico do número de créditos a atribuir a cada uma das fontes que era uma das dificuldades das versões anteriores. O algoritmo apresentado de seguida é baseado nos valores dos parâmetros Z_i, X_i, e Q_i (para $i = 1, 2, ..., N$) do

método desenvolvido no Capítulo 5 e na técnica de cálculo dos créditos consumidos, que vão servir de referência para os períodos de transmissão seguintes.

N	Número total de filas
NQ	Número de filas com células
$Fonte^{Old}$	Fonte que foi escalonada no ciclo anterior
$Fonte$	Fonte que vai ser escalonada
t_{max}	Tempo máximo de espera para escalonamento
F_{max}	Fonte que está à mais tempo à espera para ser escalonada
$credits_{min}$	Número de créditos concedidos a cada fonte
$credits_i$	Número de créditos consumidos pela fonte i
$credits_i^{acu}$	Número total de créditos consumidos pela fonte i
$tFlag$	Variável de estado que controla o primeiro teste de mudança
$flagX$	Variável de estado que controla o valor de x
$\aleph(t)$	Conjunto de filas com células no instante t
$\#\aleph(t)$	Número total de elementos do conjunto $\aleph(t)$ no instante t

/* Cálculo da próxima fonte a ser escalonada */

01. if $(\#\aleph = 0)$ **/* Inicialização */**
02. $Inicializa_param()$
03. $else$
04. $do\ for\ \forall i \in \aleph$ **/* Célula à mais tempo à espera */**
05. $x_i \leftarrow b_i - (Z_i + Q_i)$
06. $t_{max} \leftarrow MAX(t_{max}, time() - cell.time_i)\{F_{max} \leftarrow i\}$
07. if $(\#\aleph = 1)$ **/* Apenas uma fonte a transmitir */**
08. $tFlag \leftarrow 0$
09. $Fonte^{Old} \leftarrow Fonte$
10. $else$ **/* Mais do que uma fonte com células para servir */**
11. $do\ for\ \forall i \in \aleph$
12. if $(x_i >= 0)$ **/* Garante transmissão acima de SCR_i */**
13. if $(Z_i/b_i >= max \wedge credits_i <= credits_{min} \wedge flag_i = 0)$
14. $Max \leftarrow Z_i/b_i$
15. $Fonte \leftarrow i$
16. $else$
17. if $(flag_i == 0 \wedge credits_i > credits_{min})$
18. $flag_i \leftarrow 1$
19. $Fonte \leftarrow F_{max}$
20. $flag_{F_{max}} \leftarrow 0$
21. $flagX \leftarrow 0$
22. $else$ **/* x é negativo */**
23. $flag_i \leftarrow 0$
24. $Fonte \leftarrow i$
25. $flagX \leftarrow 1$

26. $if\ (Fonte\ !=\ Fonte^{Old}\ \wedge\ Fonte\ !=\ -1)$
27. $flag_{Fonte^{Old}} \leftarrow 1$
28. $Fonte^{Old} \leftarrow Fonte$
29. **/* Cálculo do novo intervalo de transmissão */**
30. $if\ (tFlag == 0)$ **Executado apenas uma vez */**
31. $tFlag \leftarrow 1$
32. $if\ (Z_{Fonte^{Old}}/b_{Fonte^{Old}} < 0.5)$
33. $interval \leftarrow MBS_{Fonte^{Old}} - credits_{Fonte^{Old}}$
34. $else$
35. $interval \leftarrow credits_{Fonte^{Old}}$
36. $else$
37. $if\ (flagX == 0)$
38. $interval \leftarrow credits_{Fonte^{Old}}$
39. $credits_{min} \leftarrow MIN(credits_{min}, interval)$
40. $credits_{Fonte^{Old}} \leftarrow 0$

/* Verificar se alguma condição falhou */

41. $if(Fonte == -1\ \wedge\ NQ\ !=\ 0)$
42. $Fonte \leftarrow F_{max}$
43. $flag_{F_{max}} \leftarrow 0$
44. $credits_{F_{max}} \leftarrow 0$

/* Escalonar célula da fonte $Fonte$ ***/**

45. $credits_{Fonte}++$
46. $credits^{acu}_{Fonte}++$
47. $Q_{ABR} \leftarrow Q_{Fonte}$

6.3.5 Resultados de simulação

Esta solução algorítmica foi testada através de simulação, mantendo as mesmas condições dos algoritmos anteriores.

Cenário 1

Nos resultados apresentados na Figura 6.18 é possível observar que a ocupação dos *buffers* das fontes (gráfico (c)) apresenta um padrão idêntico, o que mostra uma distribuição igual da largura de banda pelas duas fontes, assim como neste caso é conseguida uma melhoria da distribuição dos atrasos (gráfico (d)), relativamente aos resultados análogos da solução anterior. Nestes resultados os valores de X (Figura 6.18 (b)) mostram a forma eficiente como a largura de banda é distribuída dentro do ED, isto é, a progressão dos valores de X depende exclusivamente do número de créditos atribuídos a

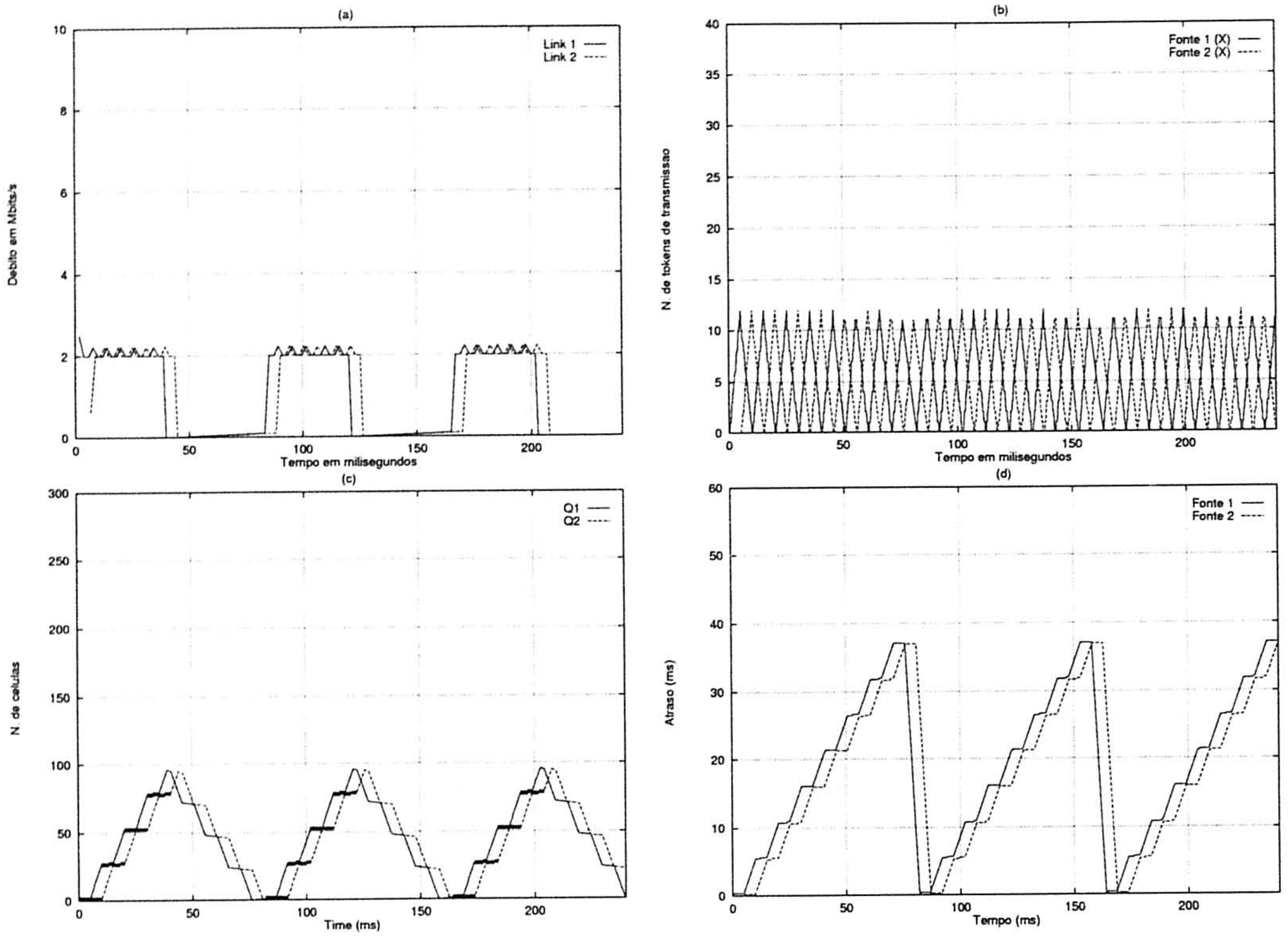

Figura 6.18: Resultados obtidos com o terceiro algoritmo no Cenário 1.

cada fluxo, garantindo que em nenhuma situação o seu valor seja negativo, o que fornece uma garantia quanto aos valores acordados aquando do controlo de admissão.

Cenário 2

Os objectivos desta simulação eram verificar o comportamento do algoritmo quanto à distribuição equitativa dos atrasos e o controlo do *backlog* quando uma fonte praticamente terminou a sua transmissão e uma segunda está apenas no seu início. Estes levaram à introdução da técnica descrita anteriormente, o que resultou na observação de atrasos máximos de 5 ms, quando o tempo de decalagem entre elas é de 37.4 ms, como apresentado na Figura 6.19. Da mesma forma, o comportamento dos valores de X são os ideais, uma vez que a troca do controlo é efectuado e os valores de X de ambas as fontes estavam bem afastados (gráfico (b)), conseguindo-se assim atingir os objectivos inicialmente propostos.

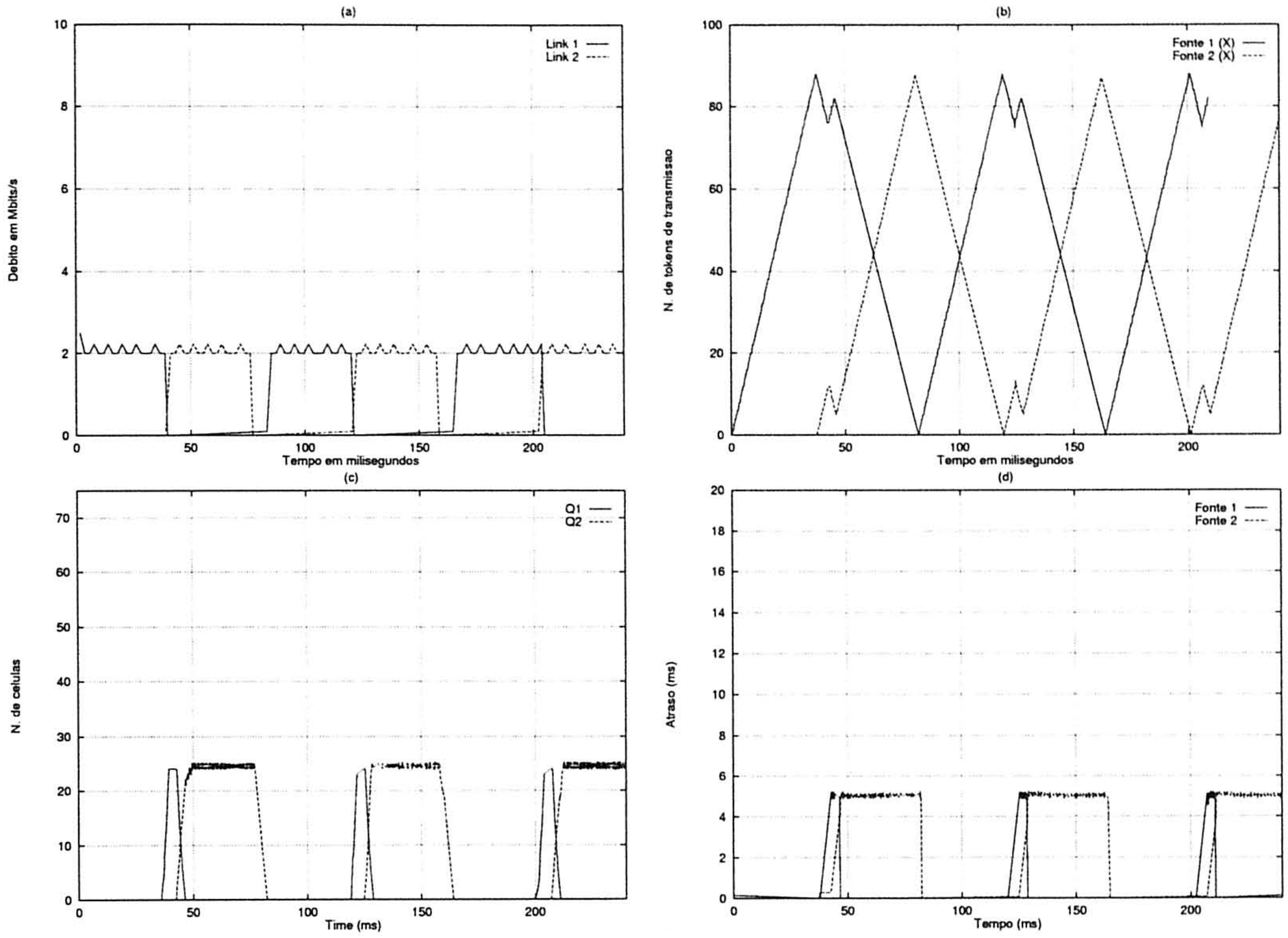

Figura 6.19: Resultados obtidos com o terceiro algoritmo no Cenário 2.

Cenário 3

Os resultados apresentados na Figura 6.20 mostram as vantagens desta solução com este tipo de cenário, isto é, os gráficos (c) e (d) indicam que tanto a ocupação dos *buffers* como a distribuição dos atrasos é a esperada.

Cenário 4

Os resultados obtidos sobre este cenário (Figura 6.21) mostram a ocupação dos *buffers* (gráfico (c)) com as devidas proporções, relativamente aos débitos de geração de tráfego de cada uma das fontes, isto é, a ocupação do *buffer* da segunda fonte é três vezes superior ao da primeira fonte; no entanto, o gráfico referente às distribuições dos atrasos (d) não apresenta essa proporção porque a segunda fonte é servida mais vezes do que a primeira.

Relativamente aos valores de X verifica-se a influência deste valor ser ou não negativo. Esta situação verifica-se porque, enquanto a Fonte 2 transmite durante 5 ms, a Fonte 1 transmite até que o valor de X da Fonte 2 seja igual a zero e então o serviço

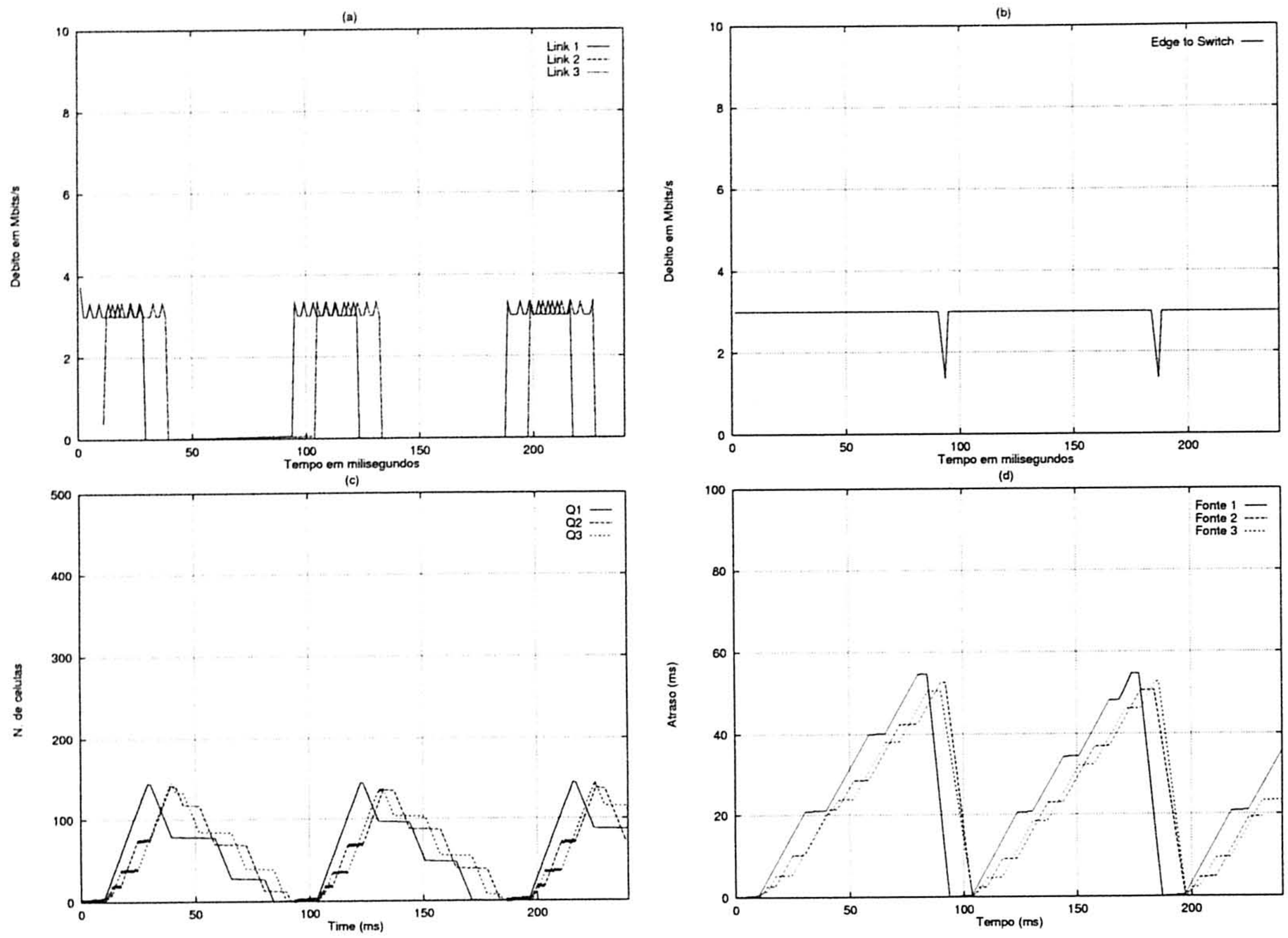

Figura 6.20: Resultados obtidos com o terceiro algoritmo no Cenário 3.

volta a ser atribuído à Fonte 2. Desta forma, a Fonte 2 é servida mais vezes do que a Fonte 1, o que resulta na distribuição esperada dos atrasos.

Cenário 5

Os resultados da Figura 6.22 mostram que a ocupação dos recursos é total e a distribuição da largura de banda é realizada de forma equitativa, mesmo quando as fontes iniciam a sua actividade em diferentes períodos de tempo. A observação dos gráficos (c) e (d) permite perceber como a distribuição da largura de banda é realizada, sem beneficiar nenhuma das fontes, independentemente do seu início de actividade. O padrão encontrado na ocupção dos *buffers* a jusante dos TBs e da consequente distribuição dos atrasos prova isso mesmo.

Os valores de X (ver gráfico (b) da Figura 6.22) servem para complementar os resultados já analisados, uma vez que apresentam um padrão idêntico à medida que a simulação evolui, isto é, têm sempre uma variação entre 0 e 35, para além de nunca serem inferiores a zero.

Estes resultados dão uma boa indicação, a par dos resultados obtidos com os

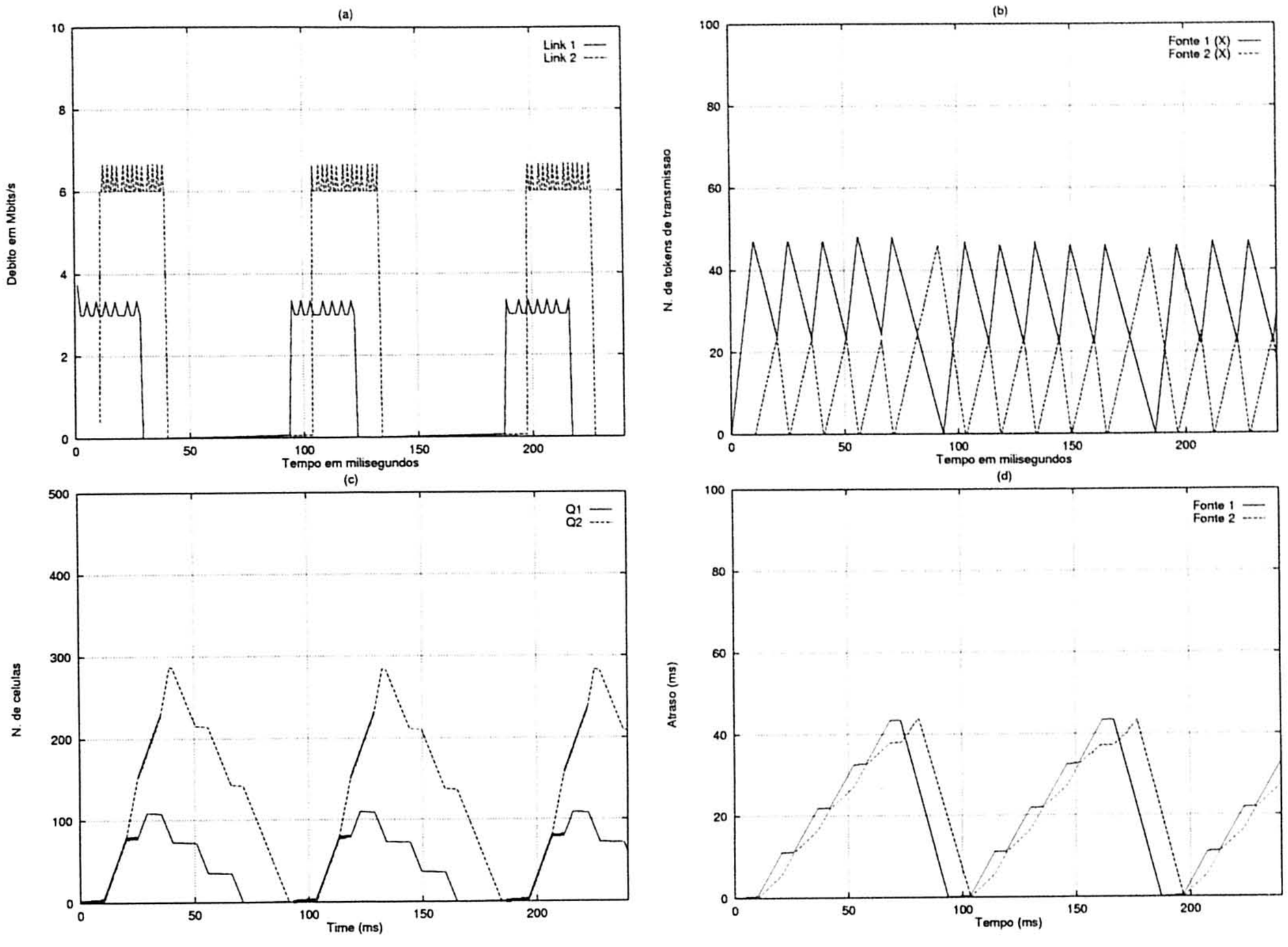

Figura 6.21: Resultados obtidos com o terceiro algoritmo no Cenário 4.

outros cenários, relativamente à forma simples e eficiente como o algoritmo, com base nos valores dos parâmetros dos TBs e da técnica de cálculo dos créditos, pode fazer uma distribuição equitativa da largura de banda e, consequentemente ter em atenção quais as fontes já beneficiadas inicialmente e a forma como as outras devem ser compensadas sem prejudicar as primeiras.

Cenário 6

Os resultados da Figura 6.23 servem apenas para confirmar o desempenho já observado relativamente ao algoritmo proposto.

6.3.6 Conclusões

Os resultados apresentados para a terceira solução algorítmica revelam um bom desempenho para o método inicialmente proposto, uma vez que os problemas encontrados nas propostas anteriores foram solucionados e os objectivos propostos foram todos alcançados, isto é, a largura de banda, a ocupação dos *buffers* e a distribuição dos atrasos

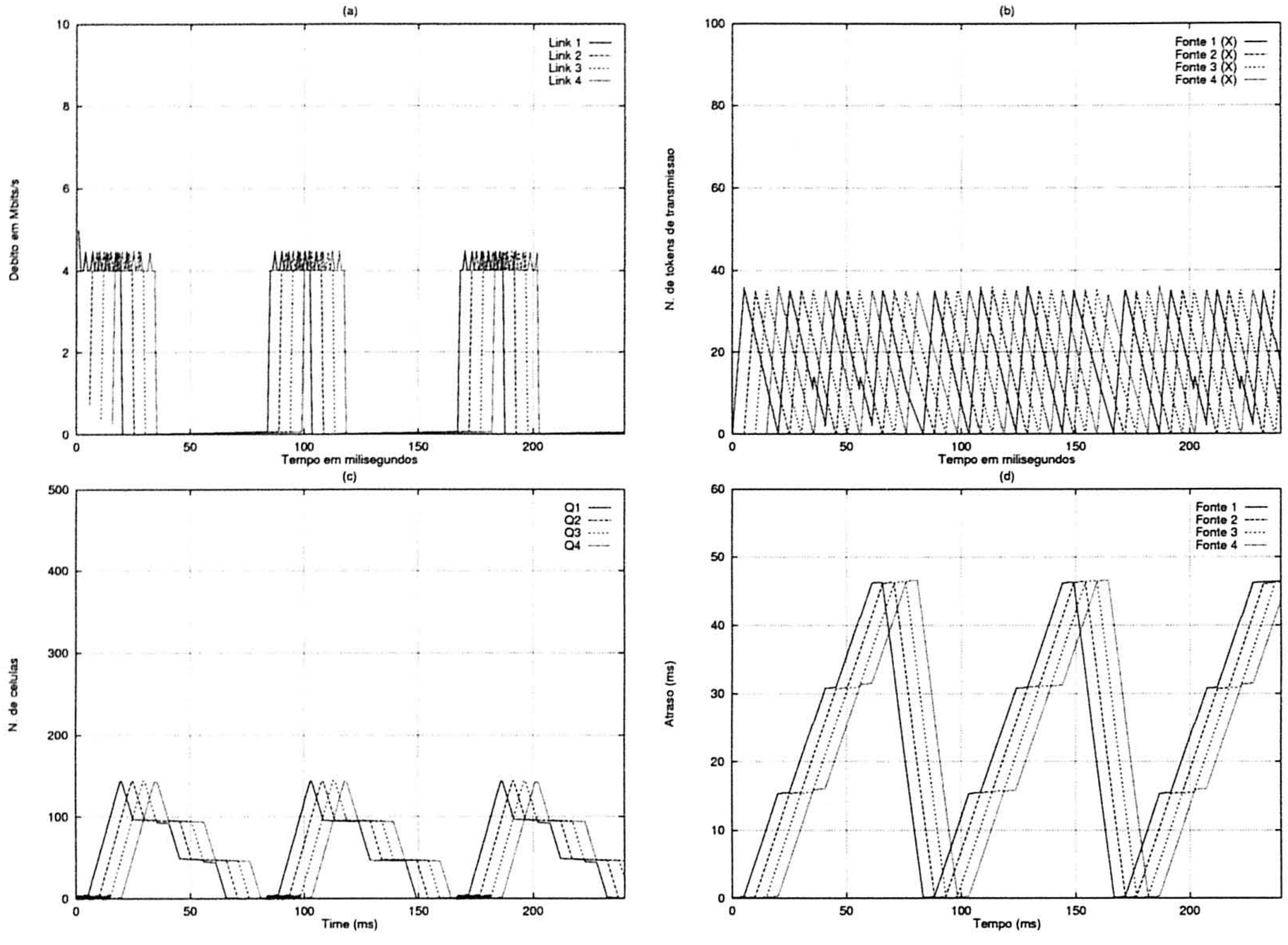

Figura 6.22: Resultados obtidos com o terceiro algoritmo no Cenário 5.

pelos fluxos em competição é realizada de forma equitativa.

6.4 Complexidade algorítmica

Com base nos algoritmos propostos é possível, através de simulação ou da implementação em situações reais, obter resultados que serão posteriormente analisados com o objectivo de darem indicações para eventuais alterações. Contudo, essas indicações poderão ter várias vertentes, nomeadamente, a reformulação do algoritmo porque os objectivos não foram alcançados, ou a necessidade de reduzir a complexidade algorítmica, ou ainda a alteração das estruturas de dados utilizadas de forma a que o algoritmo possa ser implementado. São, precisamente as duas últimas questões (complexidade algorítmica e estruturas de dados) que são objecto de discussão nesta secção, uma vez que a primeira questão foi discutida nas secções anteriores.

A complexidade algorítmica e as estruturas de dados utilizadas nas três versões algorítmicas apresentadas podem ser comparadas com a complexidade algorítmica e as estruturas de dados de algumas disciplinas de serviço apresentadas no Capítulo 4.

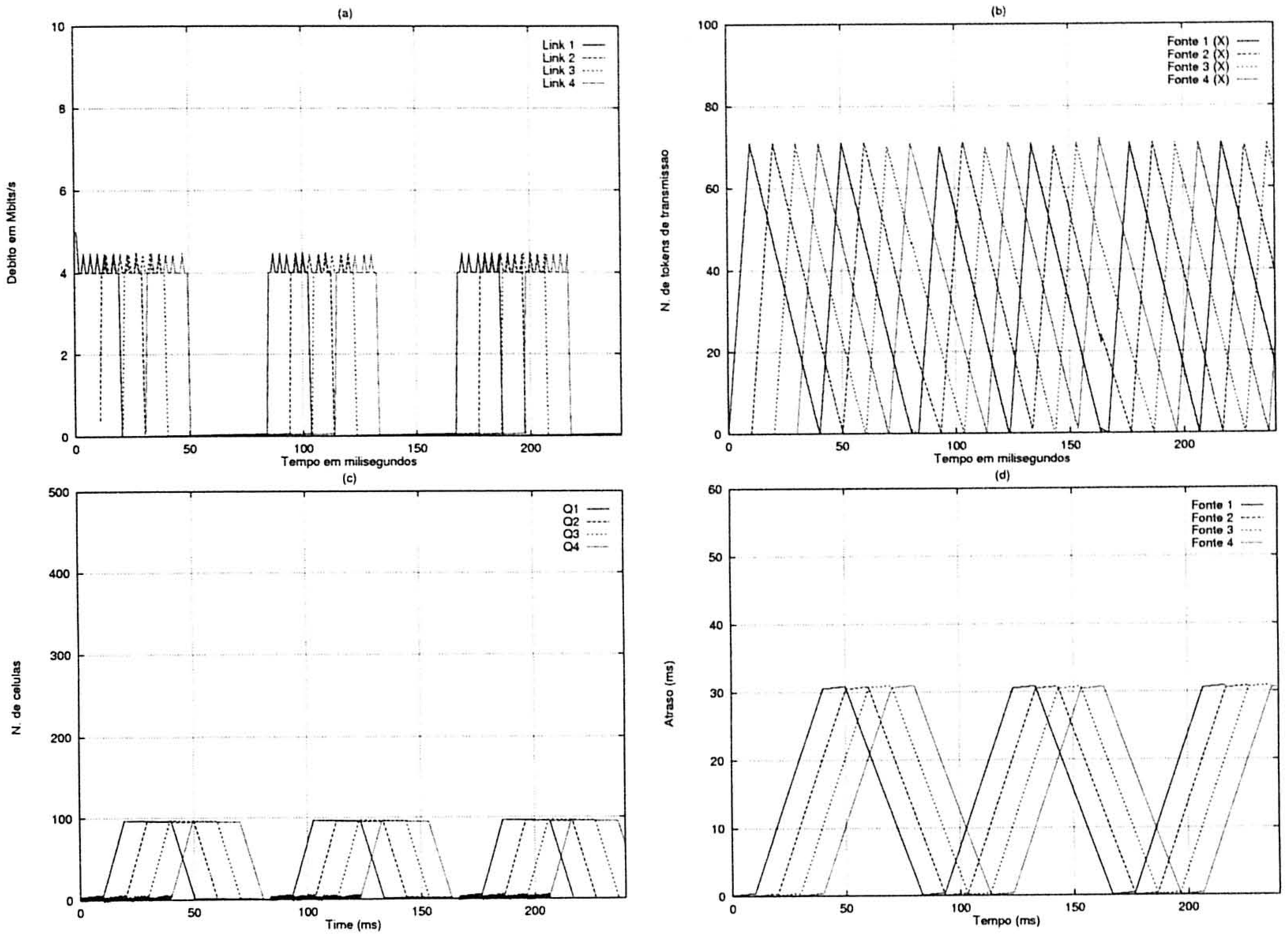

Figura 6.23: Resultados obtidos com o terceiro algoritmo no Cenário 6.

Antes de discutir as várias versões do algoritmo proposto, relativamente às questões apresentadas anteriormente, são resumidamente discutidas as questões de complexidade em relação às disciplinas de serviço mais conhecidas.

As disciplinas de serviço do tipo *WCing* têm dois custos de implementação, o primeiro refere-se ao cálculo da função tempo virtual e o segundo está relacionado com o custo de manutenção de uma fila com prioridade, baseada nos tempos virtuais calculados. Esta estrutura de dados necessita do cálculo do índice de prioridade por pacote ou célula e apresenta uma compexidade de $\bigcirc(\log n)$ [71], com n a representar o número de sessões activas. Para as disciplinas de serviço que usam o tempo real para calcular o índice de prioridade, tais como Virtual Clock e Delay-EDD, o cálculo é simples, mas para as disciplinas que usam tempos virtuais como referência, WFQ, WF^2Q e WF^2Q+, o cálculo é mais complexo, para além de necessitarem de regular os pacotes com base nos seus tempos de início ou de eligibilidade.

Tendo por base as questões discutidas verifica-se que a primeira solução algorítmica apresentada na secção 6.1 tem uma complexidade algorítmica de $\bigcirc(n)$, com n a representar o número de filas com células, e utiliza estruturas de dados que têm uma

complexidade de $\bigcirc(1)$, que comparando com as disciplinas WFQ, WF^{2}Q e WF^{2}Q+ apresenta uma complexidade mais reduzida, quer nas estruturas de dados utilizada quer na eliminação do cálculo da função de tempo virtual.

A solução apresentada na secção 6.2 é semelhante à primeira versão, no que se refere à complexidade algorítmica e à complexidade das estruturas de dados. Esta situação leva a que seja possível concluir, em termos gerais, que a segunda solução apresenta uma complexidade mais baixa que as disciplinas de referência. Contudo, é possível discutir o tamanho do algoritmo apresentado quanto à sua implementação num equipamento real. Segundo a teoria da complexidade algorítmica esse facto não é relevante desde que não existam estruturas de dados complexas (listas, árvores, filas com prioridade, etc.), execução de ciclos de instruções dentro de ciclos de instruções (deve-se evitar ter complexidades $\bigcirc(n^3)$, $\bigcirc(2^n)$ ou $\bigcirc(e^n)$), ou ainda cálculos complexos (cálculo de expressões que contenham exponenciais, por exemplo), o que não ocorre em nenhum dos casos.

A terceira e última versão apresentada, ver secção 6.3, tem a mesma complexidade algorítmica e usa as mesmas estruturas de dados da versão apresentada na secção 6.2, mas adicionalmente apresenta um número mais reduzido de instruções a executar.

Como conclusão, é possível afirmar que qualquer uma das versões algorítmicas apresentadas neste capítulo tem uma complexidade algorítmica igual ou inferior à complexidade algorítmica das disciplinas de serviço de referência (WFQ, WF^{2}Q e WF^{2}Q+), mas apresenta uma complexidade inferior relativamente às estruturas de dados, uma vez que as disciplinas de referência apresentam uma complexidade de $\bigcirc(\log n)$ e as soluções apresentadas neste capítulo têm uma complexidade de $\bigcirc(1)$.

6.5 Resumo e conclusões

Neste capítulo foram apresentadas três soluções algorítmicas baseadas em diferentes conceitos para implementar o método apresentado e desenvolvido no Capítulo 5.

A primeira solução é a mais simples do ponto de vista de implementação, mas apresenta algumas limitações, em termos de resultados, em que se pode destacar, por um lado, a não obtenção de todos os objectivos propostos e, por outro lado, é uma proposta que realiza (converge para) uma distribuição equitativa da largura de banda

pelos fluxos em competição (pode ser observado através do comportamento dos valores de X quando eles convergem todos para o mesmo valor), mas garantindo que o débito médio negociado é sempre assegurado.

Embora, os resultados anteriores não tenham sido os melhores serviram como ponto de partida para a segunda solução que se baseou na ideia de períodos de tempo com um comprimento fixo. A ideia do comprimento fixo aplica-se apenas no sentido em que, a partir do instante em que o sistema esteja num estado estável, os períodos atribuídos têm sempre o mesmo comprimento. Os resultados obtidos com esta solução são os esperados, uma vez que foram atingidos os objectivos inicialmente propostos. Contudo, a ideia de intervalo de tempo não era a melhor porque era complicado saber quando um *burst* começava e acabava, admitindo que as células que os constituem eram transmitidas sempre com o débito de pico, o que não se verifica na prática; no entanto, os resultados conseguidos eram positivos e mostravam que a ideia de atribuir períodos de comprimento fixo poderia ser a base da solução.

A última solução tem como principal preocupação resolver os problemas anteriores, mas manter a ideia dos períodos de comprimento fixo. Com este propósito decidiu-se utilizar valores discretos em substituição dos tempos, para controlar os intervalos de serviço atribuídos a cada fluxo. A solução encontrada passou por adoptar o conceito de **créditos**, o que permitiu ultrapassar o problema anterior, porque o início do *burst* é marcado pelo número de créditos consumidos por um deteminado fluxo ser zero e o fim é atingido quando o total for igual ao número de células que constituem o *burst*. Assim, quando todos os fluxos atingirem esta condição, os valores dos créditos consumidos podem voltar a ser reinicializados a zero e um novo ciclo de transmissão e controlo é iniciado.

Pode-se concluir que todos os objectivos propostos foram atingidos, uma vez que os resultados obtidos ao longo deste capítulo apresentam uma equidade na distribuição da largura de banda e nos atrasos e no controlo da ocupação dos *buffers*.

Capítulo 7

Distribuição de tráfego ABR nos Sistemas Periféricos

Com o algoritmo de escalonamento definido e com a categoria de serviço escolhida para suportar a agregação de fluxos, era necessário definir um critério para a distribuição da largura de banda pelos VCs em competição, permitindo que este tipo de VCs possam ter ou não débitos mínimos garantidos.

Na primeira secção é realizado um breve enquadramento dos assuntos que vão ser discutidos neste capítulo. Tendo-se optado por utilizar VCs ABR e como esta categoria necessita de algoritmos de controlo de fluxo e de congestionamento, é apresentado na segunda secção um conjunto de critérios para o cálculo dos pesos a utilizar na distribuição da largura de banda. Na terceira secção é realizada a avaliação e escolha do critério para o cálculo dos pesos a atribuir a cada fonte. Na quarta secção é apresentado a solução algorítmica, equanto que na quinta secção é realizada a avaliação através de simulação. Na última secção são apresentadas as conclusões.

7.1 Introdução

Com o objectivo de apresentar uma solução completa para o ED (ver Figura 7.1) a definição de técnicas de escalonamento e agregação de tráfego IP num VC ATM (Capítulos 5 e 6) são importantes para dar garantias de QoS a este tipo de tráfego, bem como conceber algoritmos para a distribuição da largura de banda entre os VCs ABR de acordo com os valores fornecidos pela rede.

Qualquer que seja o tipo de VC escolhido para a rede de núcleo, a largura de banda disponível tem de ser distribuída pelos VCs dessa classe de forma equitativa, mas com base em critérios que têm de estar de acordo com um conjunto de pressupostos definidos à partida. Com base nos algoritmos mais representativos relativamente à distribuição da largura de banda para a categoria de serviço ABR [120, 87, 86, 9, 59, 17, 18, 96, 9, 51, 107, 3, 62, 65, 64, 63, 22, 60, 67, 116, 81, 82] e nos critérios de *fairness* indicados nos documentos do ATM Forum [44, 42] é necessário encontrar uma alternativa que obedeça, por um lado a um critério de *fairness* e, por outro lado seja implementável nos EDs.

Apesar do objectivo principal ser a distribuição deste recurso dentro dos EDs resolveu-se estender esta proposta à rede ATM, isto é, implementar (por simulação) o mecanismo de distribuição e controlo nos comutadores para verificar se, em configurações com alguma complexidade, a distribuição da largura de banda era equitativa em momentos com ou sem perturbações (quando fontes com prioridade superior iniciam e terminam a sua actividade). Estes critérios vão permitir qualquer tipo de configuração (fontes com ou sem débitos mínimos garantidos), para além de nos casos particulares o modelo ser reduzido ao caso básico *max-min fairness*.

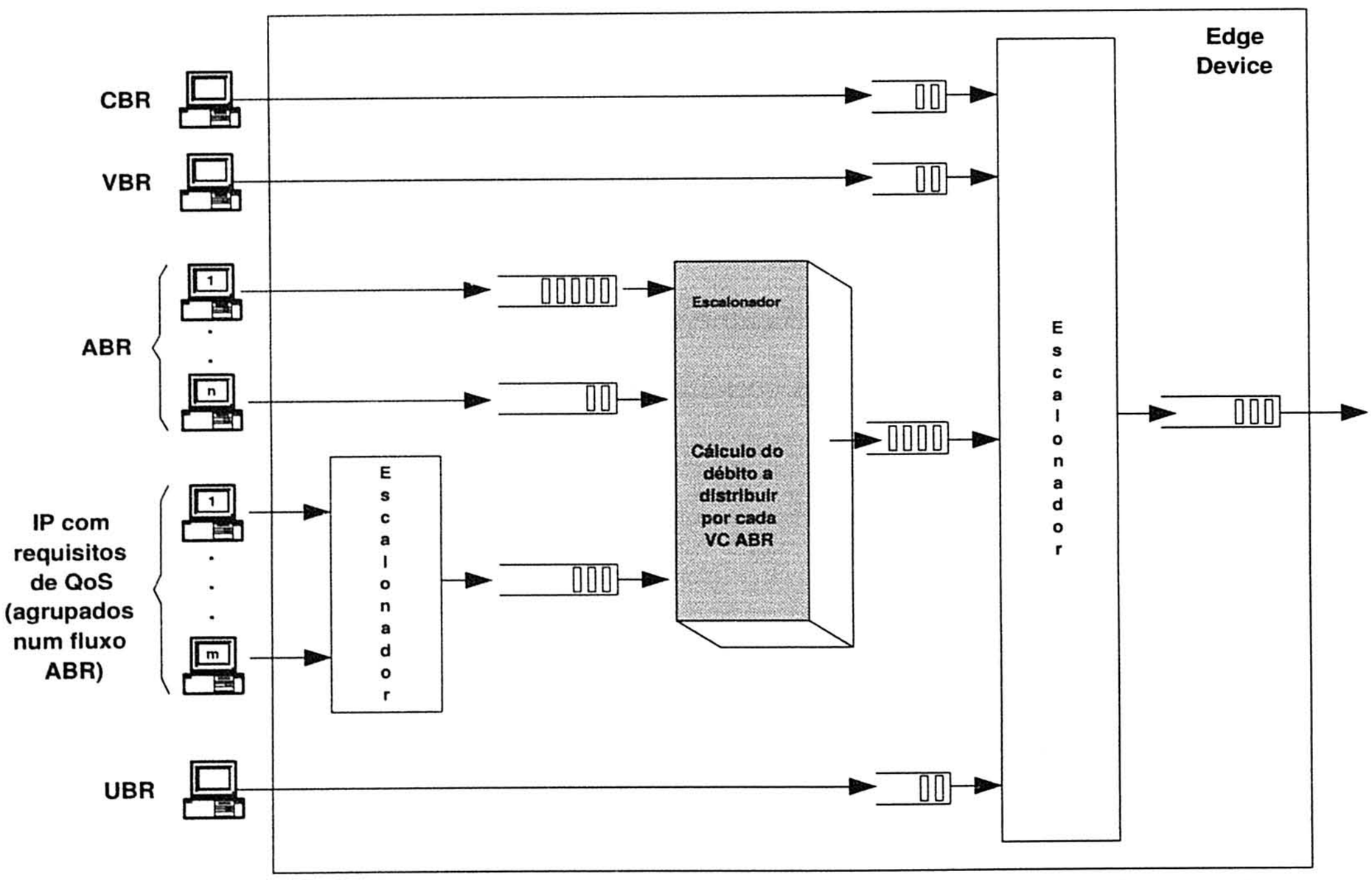

Figura 7.1: Arquitectura geral do *Edge Device* (2).

O ponto de partida para a distribuição óptima de recursos partilhados é usualmente dado pelo critério conhecido por *max-min allocation* [18], extendido posteriormente a outras definições [44, 42, 33]. Contudo, antes de apresentar qualquer proposta para esta questão serão apresentados os critérios de *fairness* indicados nos documentos do ATM Forum [44, 42] que se aplicam apenas às ligações que se encontram congestionadas e são os seguintes:

A	Largura de banda total disponível para as conexões ABR numa dada ligação;
U	Total de largura de banda usada por conexões satisfeitas, isto é, não estão a ser controladas na ligação;
$B = A - U$	Largura de banda disponível para ser partilhada pelas conexões congestionadas nessa ligação;
N	Número de conexões activas;
N'	Número de conexões activas não controladas nesta ligação;
$n = N - N'$	Número de conexões activas congestionadas nesta ligação;
MCR_i	MCR da conexão i;
$M = \sum_{i=1}^{n} MCR_i$	Largura de banda mínima garantida das conexões activas e congestionadas nesta ligação;
w_i	Peso pré-atribuído à conexão i;
B_i	Largura de banda atribuída à conexão i.

1. **Max-Min** A largura de banda B é distribuída equitativamente entre n conexões que estão congestionadas nessa ligação

$$B_i = \frac{B}{n}. \tag{7.1}$$

   Este critério apenas se aplica a casos em que as conexões são tratadas de forma idêntica e com $MCR_i = 0$.

2. **MCR mais partilha igual** A distribuição da largura de banda para uma conexão é o seu MCR, mais uma distribuição igual da largura de banda remanescente $(B - M)$

$$B_i = MCR_i + \frac{B - M}{n}. \tag{7.2}$$

   Este critério converge para o critério *Max-Min* quando todos os MCRs se aproximam de zero.

3. **MCR máximo ou partilha Max-Min** A distribuição da largura de banda para uma conexão é o seu MCR ou a partilha *Max-Min* (calculada sem ter em conta os MCRs), escolhendo o maior valor,

$$B_i = max(MCR_i, Max\text{-}Min\ share). \tag{7.3}$$

Este critério converge para o critério *Max-Min* quando os $MCRs$ se aproximam de zero. A distribuição da largura de banda, de acordo com este critério pode requerer muitas iterações para convergir para um ponto de equilíbrio, nomeadamente se o *max-min share* inicial for inferior a MCR_i.

4. **Distribuição proporcional ao MCR** A distribuição da largura de banda para uma conexão é distribuída proporcionalmente ao seu MCR

$$B_i = B \times \frac{MCR_i}{M}. \tag{7.4}$$

Este critério não se pode aplicar se existirem conexões com MCR nulos.

5. **Distribuição pesada** A distribuição da largura de banda para a conexão i é proporcional ao seu peso pré-definido, w_i

$$B_i = B \times \frac{w_i}{\sum_{j=1}^{n} w_j}. \tag{7.5}$$

O critério *Max-Min* é um caso especial com todas as conexões com pesos iguais; por outro lado, o peso pode ser definido de modo independente ou dependente do MCR (ver critério apresentado na opção 4).

6. **MCR mais distribuição pesada** A distribuição da largura de banda para a conexão i é o seu MCR, mais uma distribuição pesada da largura de banda remanescente $(B - M)$.

$$B_i = MCR_i + (B - M) \times \frac{w_i}{\sum_{j=1}^{n} w_j} \tag{7.6}$$

O *Max-Min*, o *MCR mais distribuição igual* e *Distribuição proporcional aos MCRs* são casos especiais, pondendo os pesos ser definidos de modo independente, ou dependente do MCR.

7.2 Critérios de cálculo do peso das fontes na distribuição da largura de banda remanescente

A necessidade de uma política de preços atractiva para os consumidores por parte dos operadores de telecomunicações e, em simultâneo, a obtenção de um critério de taxação e de distribuição da largura de banda remanescente levou à introdução de uma proposta que preconizasse esse objectivo.

De acordo com a ideia apresentada em [33] de que "*(...) In real world users expected a service related to the price they are paying for the service. (...)*", são apresentadas algumas soluções para a definição de um novo critério para o cálculo dos pesos que servisse de base na distribuição equitativa, tendo em atenção pressupostos relacionados com os parâmetros de tráfego e esquemas de tarifação. Antes de apresentar as várias propostas para o estabelecimento de critérios definiram-se os seguintes pressupostos:

- O custo por *Mbit/s* de reservar MCR deve ser superior ao custo por *Mbit/s* por transmitir acima de MCR.
- A fonte que contrata o MCR mais elevado deve ser favorecida na distribuição da largura de banda remanescente através da atribuição de um peso superior (obterá uma parte maior, proporcionalmente, pelo menos enquanto for controlada).
- Se $\forall_i\ MCR_i = 0$ a solução deve reduzir-se ao caso de *max-min fairness* básico.
- No critério básico não é tido em consideração o PCR_i solicitado por cada fonte.

Relativamente ao esquema de tarifação é apresentada em [33] uma solução para a política de preços, onde os autores estabeleceram um novo critério para o cálculo a aplicar a cada uma das fontes, bem como a forma como a largura de banda remanescente é distribuída.

Considerando T um intervalo de tempo de referência (para efeito de comparação), o valor C que o cliente paga por usar a rede durante esse intervalo é uma função do número de *bits* (W) que a rede transporta com sucesso,

$$C = f(W, R) \tag{7.7}$$

sendo $R = \frac{W}{T}$ o débito médio obtido.

Segundo os mesmos autores é possível assumir que $f(.)$ é não decrescente em função de W, isto é, as fontes que enviam mais *bits* não pagam menos. A função utilizada para representar o valor a cobrar é

$$C = c + w \times W + r \times R \tag{7.8}$$

em que c é o valor fixo por conexão, w o custo por *bit* e r o custo por *Mbit/s*, idênticos para todos os clientes. Contudo, como existe a necessidade de ter um débito mínimo garantido é incluido na função essa variável, então

$$C = f(W, R, M) \tag{7.9}$$

em que M representa o MCR. No entanto, é necessário indicar que quem negociar um valor superior de M deve ter um custo superior e uma função possível é

$$C = c + w \times W + r \times R + m \times M \tag{7.10}$$

em que m é o preço por *Mbit/s* de MCR. De facto, o cliente paga $r + m$ por *Mbit/s* até atingir M e depois paga apenas r por *Mbit/s* pela largura de banda extra obtida, para além do seu MCR negociado (no caso de $R > M$).

A função (7.10) pode ser rescrita da seguinte forma,

$$C = c + w \times W + r \times \Delta R + (r + m) \times M \tag{7.11}$$

com ΔR a representar a fracção de largura de banda que a fonte vai receber acima do seu MCR. Representando $\alpha = m + r$ e $\beta = r$ e, como o débito a distribuir às fontes é

$$R = M + \Delta R \tag{7.12}$$

então a função de custo é

$$C = c + w \times W + \alpha \times M + \beta \times \Delta R. \tag{7.13}$$

7.2.1 1° Critério

Uma primeira hipótese que se considerou no cálculo dos pesos foi a relação existente entre o par débito mínimo garantido/débito máximo desejado (MCR_i, PCR_i) e o débito

máximo desejado (PCR_i). De acordo com os pressupostos definidos o peso poderá ser tanto maior quanto maior for MCR_i, por outro lado a fonte poderá ser igualmente beneficiada se o excesso ($\Delta_i = PCR_i - MCR_i$, ver Figura 7.2) for relativamente pequeno quando comparado com MCR_i (ou, de forma mais ou menos equivalente com PCR_i), isto é, a maior parte da largura de banda pretendida é paga como largura de banda garantida (MCR_i).

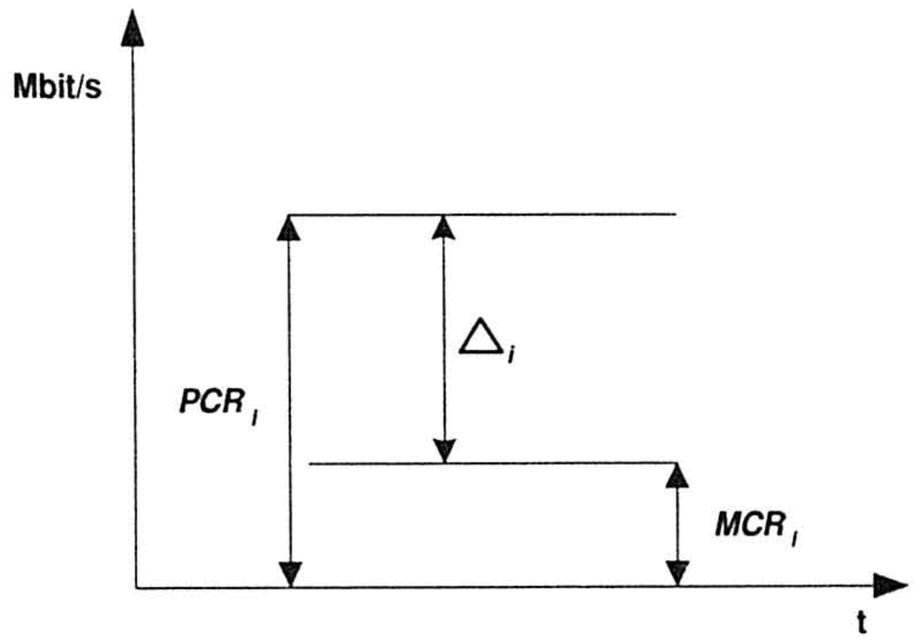

Figura 7.2: Informação utilizada no cálculo do peso de cada fonte.

Definindo

$$\phi_i = \frac{PCR_i - MCR_i}{PCR_i} \tag{7.14}$$

$$= 1 - \frac{MCR_i}{PCR_i} \tag{7.15}$$

e

$$w_i = \frac{1}{\phi_i} \tag{7.16}$$

$$= \frac{PCR_i}{PCR_i - MCR_i} \tag{7.17}$$

$$= \frac{1}{1 - \frac{MCR_i}{PCR_i}} \tag{7.18}$$

Para avaliar este critério convirá verificar o valor dos pesos em casos particulares. Quando $\forall_i \; MCR_i = 0$, o valor de $w_i = 1$ o que é uma vantagem porque inclui o caso básico e permite ter na mesma configuração fontes com $MCR_i = 0$ e com fontes $MCR_i \neq 0$. No outro extremo, isto é, $MCR_i = PCR_i$ o valor de $w_i = \infty$, o que se apresenta como uma dificuldade, mas este caso corresponde ao de uma fonte que

não necessita de largura de banda acima de MCR, logo não é contabilizada para a distribuição da banda remanescente. Por outro lado se PCR_i é superior a MCR_i, mas muito próximo, um peso muito elevado significa apenas que a fonte obterá praticamente a largura de banda pretendida.

Apesar das vantagens apresentadas este critério tem alguns problemas, nomeadamente a sua dependência do valor de PCR_i, que é um valor que a fonte pode variar dinamicamente. Outra das desvantagens é o facto de o critério não reflectir no custo a solicitação do PCR_i; no entanto solicitar um PCR_i elevado resultaria num peso menor. No caso básico ($MCR_i = 0$, $w_i = 1$), o PCR_i não intervém no critério *max-min fairness* associado a um peso; intervém apenas para decidir se uma fonte está satisfeita (não controlada) ou não satisfeita (controlada), mas não serve para definir um maior ou menor grau de insatisfação que se possa reflectir no peso.

Para ajudar a perceber a forma como os valores são obtidos é apresentado um exemplo de uma configuração simples com duas fontes. É admitido uma configuração duas fontes, com uma largura de banda disponível de 55 *Mbit/s* ($B = 55$ *Mbit/s*), com um débito mínimo garantido de $MCR_i = \{30, 10\}$, um débito máximo desejado de $PCR_i = \{50, 50\}$ e uma largura de banda remanescente de 15 *Mbit/s* ($B - \sum_i MCR_i$). Aplicando a expressão (7.14) e (7.16) obtém-se os seguintes resultados $\phi_1 = 0.6$ e $\phi_2 = 0.8$ com os pesos $w_1 = 2.5$ e $w_2 = 1.25$, mas como $w_1 + w_2 = 3.75$ é necessário fazer a sua normalização, então $p_1 = \frac{w_1}{w_1+w_2} = 0.67$ e $p_2 = \frac{w_2}{w_1+w_2} = 0.33$. Com os pesos calculados é possível fazer a distribuição da largura de banda remanescentes pelas fontes através da expressão (7.63), obtendo-se $B_1 = 40$ e $B_2 = 15$ *Mbit/s*. Quanto ao cálculo do custo (ver (7.13)) é possível admitir que o custo por *Mbit/s* de MCR_i é α e o custo por *Mbit/s* para a banda remanescente é β, então a Fonte 1 (à parte das componentes $c + w \times W$) vai ser taxada de $C_1 = \alpha \times 30 + \beta \times 10$ e a Fonte 2 de $C_2 = \alpha \times 10 + \beta \times 5$, logo o custo por *Mbit/s* (total) transmitido é $\frac{C_1}{B_1} = \frac{\alpha \times 30 + \beta \times 10}{40} = \alpha \times 0.75 + \beta \times 0.25$ e $\frac{C_2}{B_2} = \frac{\alpha \times 10 + \beta \times 5}{15} = \alpha \times 0.67 + \beta \times 0.33$. A Fonte 1 beneficia igualmente do facto de o custo fixo estar agora associado a um débito maior.

Com base na solução anterior é apresentada uma variante no cálculo do peso, isto é, os pesos são calculados através do seu complementor. Assim,

$$w_i = 1 - \phi_i \tag{7.19}$$

$$= 1 - \frac{PCR_i - MCR_i}{PCR_i} \quad (7.20)$$

$$= \frac{MCR_i}{PCR_i} \quad (7.21)$$

Analisando os mesmos casos particulares verifica-se que quando $MCR_i = 0$ o valor de $w_i = 0$ o que impossibilita desde logo a sua aplicação em configurações que sejam constituídas por fontes com $MCR_i = 0$ e fontes com $MCR_i \neq 0$. Para o caso em que $MCR_i = PCR_i$, o valor de $w_i = 1$ o que significa que a fonte está completamente satisfeita e por isso não entra na distribuição da banda remanescente.

Antes de passar à próxima solução é possível ainda especular em torno de uma segunda variante (ou alternativa) da definição de ϕ_i, isto é, não usar o valor de PCR_i como referência, mas o valor de MCR_i, ou seja,

$$\phi_i' = \frac{PCR_i - MCR_i}{MCR_i} \quad (7.22)$$

$$= \frac{PCR_i}{MCR_i} - 1 \quad (7.23)$$

e

$$w_i' = \frac{1}{\phi_i'} \quad (7.24)$$

$$= \frac{MCR_i}{PCR_i - MCR_i} \quad (7.25)$$

$$= \frac{1}{\frac{PCR_i}{MCR_i} - 1} \quad (7.26)$$

Esta solução apresenta um problema quando $MCR_i = 0$, uma vez que $w_i' = 0$ tornando esta forma de calcular os pesos não aplicável a configurações constituídas por fontes com MCRs nulos e não nulos.

7.2.2 2º Critério

A ideia apresentada na solução anterior é interessante, mas não permite uma intervenção directa do operador na atribuição dos pesos às fontes em competição.

A distribuição da largura de banda remanescente pelas fontes em competição será realizada com base em dois critérios (fonte que apresenta o maior MCR e a fonte que fica satisfeita mais depressa) e em pesos administrativamente atribuídos para, dependendo da

política utilizada pelo operador, decidir qual dos critérios deve ter mais preponderância no peso final.

O primeiro critério tem por base os pressupostos iniciais. A forma de materializar este critério é usar o valor mínimo garantido negociado inicialmente com o operador, isto é, a distribuição ser realizada em função do MCR de cada uma das fontes. Este critério é baseado no critério (7.4) do TM 4.1 ATM Forum [42] e representado por

$$w_i^1 = \frac{MCR_i}{\sum_{j=1}^{n} MCR_j} \tag{7.27}$$

em que w_i^1 representa uma parte do peso atribuído à fonte i na distribuição da largura de banda remanescente. Desta forma é possível perceber que quanto maior for o valor de MCR maior será a largura de banda que a fonte vai receber. Sendo este o critério definido para a distribuição da largura de banda pelos VCs em competição, nada de novo estaria a ser proposto porque este é um dos critérios de *fairness* apresentado na literatura.

Para complementar o critério anterior foi definido um segundo critério considerado na solução anterior (situação nunca antes considerada, pelo menos na literatura consultada), que é o intervalo de satisfação para cada uma das fontes, isto é, qual o valor da diferença entre o débito mínimo garantido e o débito máximo desejado (Δ_i, ver Figura 7.2). A ideia de considerar este intervalo impõe-se porque podem existir fontes com um débito de pico (máximo desejado) muito pequeno e que possam ser totalmente satisfeitas e, por isso, devem ser de alguma forma compensadas na atribuição da largura de banda e, consequentemente, no custo que lhes será imputado. Esta consideração tem subjacente a ideia de fazer com que *os cliente peçam o que de facto necessitam.* Assim a distribuição da largura de banda pelas fontes em competição é realizada de forma inversamente proporcional ao valor de Δ_i, isto é, se uma fonte apresenta um intervalo muito pequeno vai receber mais largura de banda do que uma que tem um intervalo grande, de forma a seguir o princípio enunciado.

A obtenção do valor do segundo critério obriga a alguns cálculos adicionais (simples) para se conseguir saber qual o peso atribuído de facto; *então se a distribuição da largura de banda fosse* em função de Δ_i, o cálculo do peso seria

$$\phi_i = \frac{PCR_i - MCR_i}{\sum_{j=1}^{N}(PCR_j - MCR_j)} \tag{7.28}$$

no entanto, como se deseja beneficiar as que apresentam os intervalos mais pequenos é necessário calcular o complemento de cada um dos valores obtidos, assim

$$p_i^b = 1 - \phi_i \tag{7.29}$$

Como o total dos complementos é sempre $N-1$ (onde N representa o número total de fontes em competição) é necessário realizar um novo cálculo dos pesos, mas agora com o novo total $(N-1)$, logo os pesos a atribuir a cada uma das fontes são

$$w_i^2 = \frac{p_i^b}{N-1}, \tag{7.30}$$

substituindo (7.28) em (7.29) e depois (7.29) em (7.30) obtém-se a seguinte expressão

$$w_i^2 = \frac{1}{N-1} \times \left(1 - \frac{PCR_i - MCR_i}{\sum_{j=1}^{N}(PCR_j - MCR_j)}\right) \tag{7.31}$$

A partir do instante em que são conhecidos os valores obtidos através de cada um dos critérios de distribuição da largura de banda é necessário calcular o valor final que contemple os dois critérios. Dado que existem dois critérios é necessário fazer a sua combinação através da introdução de dois factores (φ e ω, escolhidos administrativamente[1]) que vão decidir qual dos dois é que que terá mais preponderância; assim o peso na distribuição para cada uma das fontes é

$$p_i = \frac{\varphi \times w_i^1 + \omega \times w_i^2}{\varphi + \omega} \tag{7.32}$$

Estando definida a forma como o peso é calculado para a distribuição da largura de banda remanescente pelas fontes em competição é necessário avaliar as consequências para o caso particular de $MCR_i = 0$. Para este caso o primeiro elemento do peso

[1] O critério utilizado na atribuição destes valores é meramente indicativo, uma vez que é precisamente nesta situação que se verifica a intervenção directa do operador.

será $w_i^1 = 0$, o que implicará a aplicação da expressão final (7.32) apenas com a preponderância do segundo ramo ($p_i = \frac{\omega \times w_i^2}{\varphi + \omega}$), concluindo-se que este critério pode ser aplicado a configurações constituídas por fontes com MCRs nulos e fontes com MCRs não nulos.

Esta solução apresenta algumas vantagens em relação à anterior, nomeadamente dá a possibilidade ao operador de ter uma intervenção directa sobre a forma de cálculo do peso considerando, por um lado a fonte que apresenta o maior MCR e, por outro lado o intervalo de satisfação mais pequeno. Contudo tem uma dependência directa do valor de PCR_i (apenas no segundo elemento da proposta), bem como não reflecte no custo a sua solicitação.

É possível fazer a verificação desta proposta através de uma variante no cálculo do peso, isto é, alternativamente ao utilizado é possível testar os mesmos valores, mas desta vez com os pesos calculados através do seu inverso

$$p_i^b = \frac{1}{\phi_i} \tag{7.33}$$

$$= \frac{1}{\frac{PCR_i - MCR_i}{\sum_{j=1}^{N}(PCR_j - MCR_j)}} \tag{7.34}$$

$$= \frac{\sum_{j=1}^{N}(PCR_j - MCR_j)}{PCR_i - MCR_i} \tag{7.35}$$

Para o caso particular $\forall_i\ MCR_i = 0$ o valor de $w_i = \frac{\sum_{j=1}^{N} PCR_j}{PCR_i}$ resulta em pesos que beneficiam as fontes com PCR_i menores, logo são aquelas que "teoricamente" ficam satisfeitas mais cedo; sendo assim é possível ter fontes com $MCR_i = 0$ e fontes com $MCR_i \neq 0$ na mesma configuração.

7.2.3 3º Critério

Com o objectivo de eliminar os problemas detectados nas soluções anteriores, nomeadamente pesos a tender para infinito ($w_i = \infty$) quando $MCR_i = PCR_i$, $w_i = 0$ quando $MCR_i = 0$ e a dependência relativamente a PCR_i (o que não acontece na definição básica do *max-min fairness*) é realizada uma proposta dependente apenas do valor mínimo garantido e da capacidade total da ligação.

Para manter a ideia de que a capacidade remanescente é dividida equitativamente, qualquer que seja o valor de PCR_i declarado (que apenas é usado para decidir se a fonte está satisfeita ou não, sem entrar em conta se está muito ou pouco satisfeita), propõe-se considerar o mesmo princípio adaptado ao facto de que fontes que reservam MCR maior "necessitam" de uma banda remanescente menor, pelo que devem ser beneficiadas por esse facto.

Designa-se por AR (*Access Rate*) a capacidade da ligação de forma a eliminar o problema da declaração do PCR_i. Definindo

$$\phi_i = \frac{AR - MCR_i}{AR} \tag{7.36}$$

$$= 1 - \frac{MCR_i}{AR} \tag{7.37}$$

e

$$w_i = \frac{1}{\phi_i} \tag{7.38}$$

$$= \frac{AR}{AR - MCR_i} \tag{7.39}$$

$$= \frac{1}{1 - \frac{MCR_i}{AR}} \tag{7.40}$$

Quando $\forall_i \; MCR_i = 0$, o valor de $w_i = 1$ e portanto inclui como caso particular o caso básico.

Fazendo a verificação desta solução através de uma variante no cálculo do peso, isto é, alternativamente ao utilizado é possível testar os mesmos valores, mas desta vez com os pesos calculados através do seu complemento, tem-se

$$w_i = 1 - \phi_i \tag{7.41}$$

$$= 1 - \frac{AR - MCR_i}{AR} \tag{7.42}$$

$$= \frac{MCR_i}{AR} \tag{7.43}$$

Nesta variante para o caso particular $\forall_i \; MCR_i = 0$, o peso é $w_i = 0$, o que implica a sua não aplicabilidade a configurações constituídas por fontes com $MCR_i = 0$ e $MCR_i \neq 0$.

Contudo, é possível pensar em realizar uma correcção adicional porque enquanto $AR - MCR_i$ corresponde à "expectativa" da fonte (no máximo), sem considerar as restantes (e um pequeno MCR_i, corresponde a uma expectativa maior, mas de acordo com o critério adaptado, a um peso menor) deve-se considerar que o que está potencialmente disponível para partilhar é $AR - \sum_i MCR_i$, então

$$\phi_i = \frac{AR - MCR_i}{AR - \sum_j MCR_j} \tag{7.44}$$

e

$$w_i = \frac{1}{\phi_i} \tag{7.45}$$

$$= \frac{AR - \sum_j MCR_j}{AR - MCR_i} \tag{7.46}$$

Relativamente aos casos particulares é possível verificar que $w_i = 1$ quando $\forall_i$ $MCR_i = 0$, o que resulta na aplicação ao caso básico.

De forma análoga às soluções anteriores é necessário testar se o critério complementar é aplicável, então

$$w_i = 1 - \phi_i \tag{7.47}$$

$$= 1 - \frac{AR - MCR_i}{AR - \sum_j MCR_j} \tag{7.48}$$

verificando-se que não é possível aplicar esta variante (7.47) devido ao facto do valor do peso ser negativo.

7.2.4 4º Critério

Uma extensão natural da proposta anterior seria considerar uma solução que tivesse em conta não a capacidade remanescente máxima (em relação ao AR) mas sim a real, isto é, a largura de banda disponível para a categoria ABR (B, que inclui MCR, com $B \geq \sum_j MCR_j$), logo

$$\phi_i = \frac{B - MCR_i}{B - \sum_j MCR_j} \tag{7.49}$$

e

$$w_i = \frac{1}{\phi_i} \tag{7.50}$$

$$= \frac{B - \sum_j MCR_j}{B - MCR_i} \tag{7.51}$$

Quando $\forall_i\ MCR_i = 0$ resulta um peso de $w_i = 1$ e, de forma similar à solução anterior também resulta na aplicação do caso básico; no entanto, existe uma dificuldade adicional que é a variabilidade da largura de banda distribuída pela categoria de serviço ABR, isto é, os pesos variam conforme a largura de banda disponível, o que levanta dificuldades adicionais.

Segundo o mesmo princípio de testar a complementariedade da solução anterior (7.50) tem-se

$$w_i = 1 - \phi_i \tag{7.52}$$

$$= 1 - \frac{B - \sum_j MCR_j}{B - MCR_i} \tag{7.53}$$

e verifica-se que não é possível aplicar a segunda variante (7.52) devido ao facto do valor do peso ser negativo.

7.2.5 5º Critério

Esta solução elimina a variação da largura de banda disponível, uma vez que não tem em conta a largura de banda atribuída à categoria ABR (B, que inclui MCR, com $B \geq \sum_j MCR_j$), mas o total da largura de banda contratada inicialmente, logo

$$\phi_i = \frac{\sum_j MCR_j - MCR_i}{\sum_j MCR_j} \tag{7.54}$$

e

$$w_i = \frac{1}{\phi_i} \tag{7.55}$$

$$= \frac{\sum_j MCR_j}{\sum_j MCR_j - MCR_i} \tag{7.56}$$

$$= \frac{1}{1 - \frac{MCR_i}{\sum_j MCR_j}} \tag{7.57}$$

Quando $\forall_i\ MCR_i = 0$ resulta um peso de $w_i = 1$ (de forma similar à solução anterior também resulta na aplicação do caso básico) e quando $MCR_i = \sum_j MCR_j$ resulta num peso de $w_i = \infty$. Esta situação conduz a um possível "*starvation*" das restantes fontes enquanto a única fonte com $MCR_i \neq 0$ não se encontrar satisfeita, isto é, terá toda a largura de banda disponível até atingir o seu PCR_i.

De forma análoga à solução anterior (7.55) é necessário analisar o critério proposto através de uma variante do cálculo dos pesos, assim

$$w_i = 1 - \phi_i \tag{7.58}$$

$$= 1 - \frac{\sum_j MCR_j - MCR_i}{\sum_j MCR_j} \tag{7.59}$$

$$= \frac{MCR_i}{\sum_j MCR_j} \tag{7.60}$$

Quando $\forall_i\ MCR_i = 0$ resulta num peso $w_i = \infty$ e quando $\sum_j MCR_j = MCR_i$ o peso $w_i = 1$, resultando na aplicação do caso básico.

7.2.6 6° Critério

Com o objectivo de resolver os problemas da solução anterior e partindo da relação (7.55) e admitindo que $\frac{MCR_i}{\sum_j MCR_j} \ll 1$, a expressão equivalente é

$$1 + \frac{MCR_i}{\sum_j MCR_j} \tag{7.61}$$

Com base na relação (7.61) e na ideia apresentada no **2° Critério** é possível encontrar uma solução para o problema de ter configurações constituídas por fontes com e sem MCR_i nulos.

A ideia apresentada na segunda solução baseia-se na utilização de um peso constituído por duas partes, isto é, o primeiro termo traduz pesos idênticos no caso básico e o segundo termo produz o efeito do MCR_i na distribuição. Desta forma, é possível a utilização de uma expressão para o cálculo dos pesos da seguinte forma

$$w_i = k + \frac{MCR_i}{\sum_j MCR_j} \tag{7.62}$$

em que k representa um valor fixo para a distribuição por todas as fontes que estejam a disputar a largura de banda, sendo a escolha desse valor da responsabilidade do operador; por exemplo, se pretender um $k = 1$ e tiver uma configuração com N fontes, em que apenas uma apresenta um $MCR_i \neq 0$ essa fonte irá ter um peso que será o dobro de qualquer uma das outras; por outro lado, se o valor escolhido é $k = \frac{1}{N}$ a mesma fonte terá $N+1$ vezes largura de banda do que as outras fontes em relação ao valor base ($B \times \frac{1}{N}$), ou no caso dos MCR_i serem todos iguais, isto é, $MCR_i = \frac{\sum_j MCR_j}{N}$, o segundo termo é igual a $\frac{1}{N}$ e todas as fontes receberiam $B \times \frac{2}{N}$.

Quando $\forall_i\ MCR_i = 0$ resulta um peso de $w_i = k$ para todas as fontes, reduzindo-se na aplicação do caso básico e quando os MCR_i são todos iguais resulta num peso $w_i = k + \frac{1}{N}$.

7.3 Escolha do critério para o cálculo dos pesos

Após a apresentação das várias propostas é necessário escolher o critério que apresenta o melhor compromisso entre as várias propriedades para o cálculo dos pesos a atribuir às fontes. O critério a usar será escolhido tendo por base os pressupostos definidos inicialmente, as propriedades de cada um dos critérios e um exemplo numérico.

7.3.1 Análise comparativa das várias soluções

Para uma análise e escolha adequada da solução a utilizar no cálculo dos pesos é apresentada a Tabela 7.1, que resume as propriedades de cada uma das soluções. Nesta tabela é possível observar, para cada solução, qual o termo de referência (o respectivo valor ou o parâmetro usado) sobre o qual os pesos serão calculados, a sua variabilidade, os casos particulares e a que tipo de configurações é que cada uma das soluções pode

Tabela 7.1: Resumo das propriedades dos vários critérios.

Critério	Variante	Termo de referência		Casos particulares		Configuração com
		Valor	Variável	$\forall_i\ MCR_i = 0$	Outros	
1	V1	PCR^*	Sim	$w_i = 1$	$MCR_i = PCR_i$ $w_i = \infty$	$MCR_i = 0$ $MCR_i \neq 0$
	V2	PCR^*	Sim	$w_i = 0$	$MCR_i = PCR_i$ $w_i = 1$	– $MCR_i \neq 0$
2	V1	PCR^*	Sim	$w_i = \frac{\sum_j PCR_j}{PCR_i}$	–	$MCR_i = 0$ $MCR_i \neq 0$
	V2	PCR^*	Sim	$w_i = \frac{\omega \times w_i^2}{\varphi + \omega}$	–	$MCR_i = 0$ $MCR_i \neq 0$
3 a)	V1	AR	Não	$w_i = 1$	–	$MCR_i = 0$ $MCR_i \neq 0$
	V2	AR	Não	$w_i = 0$	–	– $MCR_i \neq 0$
3 b)	V1	$AR - MCR$	Não	$w_i = 1$	–	$MCR_i = 0$ $MCR_i \neq 0$
4	V1	$B - MCR$	Sim	$w_i = 1$	–	$MCR_i = 0$ $MCR_i \neq 0$
5	V1	MCR	Não	$w_i = 1$	$MCR_i = \sum_j MCR_j$ $w_i = \infty$	$MCR_i = 0$ $MCR_i \neq 0$
	V2	MCR	Não	$w_i = \infty$	$MCR_i = \sum_j MCR_j$ $w_i = 1$	– $MCR_i \neq 0$
6	V1	MCR	Não	$w_i = k$	MCR_i iguais $w_i = \frac{k \times N + 1}{N}$	$MCR_i = 0$ $MCR_i \neq 0$

Variante: V1 = $\frac{1}{\phi_i}$, V2 = $1 - \phi_i$

* Não se reflete no custo a sua solicitação

ser aplicável.

Como foi discutido ao longo da apresentação das soluções, um dos factores importantes e que se foi eliminando à medida que novas propostas foram apresentadas foi o termo de referência e a sua variabilidade. Da observação da Tabela 7.1 é possível concluir que as duas primeiras soluções (incluindo as suas variantes) estariam eliminadas à partida porque fazem depender o seu termo de referência de um valor solicitado pela fonte e com a agravante que essa operação não é taxada. As soluções seguintes já apresentam valores que não dependem de qualquer valor solicitado pelas fontes (AR, $AR - MCR$, $B - MCR$ e MCR) e praticamente todos não são variáveis com a excepção do valor $B - MCR$, uma vez que o valor de B representa a largura de banda distribuída à categoria de serviço ABR e, que por definição, pode variar a qualquer momento.

A preocupação seguinte é analisar os casos particulares, de modo a conseguir soluções que não excluam nenhum tipo de configuração, isto é, permitir configurações constituídas por fontes com $MCR_i = 0$ e/ou $MCR_i \neq 0$. Esta propriedade leva à exclusão de mais algumas soluções, nomeadamente a solução **1 V2**, a solução **3 a) V2** e a solução **5 V2** devido ao facto de suportarem apenas configurações com $MCR_i \neq 0$.

Com base na discussão realizada, as soluções que apresentam os melhores compromissos são a **3 b) V1**, a **5 V1** e a **6 V1** porque permitem configurações com MCR_i nulos e não nulos e o seu termo de referência não varia; no entanto, a solução **3 b) V1** tem um termo de referência que não tem "directamente" a ver com o que de facto os VCs ABR necessitam e a solução **5 V1** implica que as fontes que têm o seu $MCR_i = 0$ fiquem sem largura da banda atribuída, até que a fonte com $MCR_i \neq 0$ tenha atingido o seu PCR_i. Embora os critérios **3 b) V1** e **5 V1** apresentam um bom compromiso têm situações que os colocam em desvantagem, quando comparados com o critério **6 V1**. Dado que a escolha deve incidir sobre a solução **6 V1**, o exemplo numérico apresentado de seguida servirá como informação complementar para a escolha final.

7.3.2 Exemplo numérico

Estando definido o conjunto de soluções para o cálculo dos pesos é apresentado um exemplo com diferentes valores de MCR e PCR ($MCR_i = \{5, 20, 10\}$ *Mbit/s* e $PCR = \{15, 50, 25\}$ *Mbit/s*), para que seja possível observar em conjunto os diferentes resultados obtidos por cada uma das soluções; no entanto é necessário indicar que qualquer outra combinação de valores de PCR e MCR poderá dar resultados completamente diferentes dos apresentados, por isso estes valores não servem como critério para a escolha de uma das soluções, podendo apenas contribuir com informação adicional para a escolha do mesmo.

Da observação da Tabela 7.2, a primeira conclusão que é possível extrair é a confirmação dos pressupostos assumidos durante a definição dos critérios, isto é, como a Fonte 1 apresenta o menor MCR logo seria a mais penalizada, mas como também apresenta o menor Δ_1, essa penalização é relativamente minimizada em função da solução **2 V1**; no entanto, a fonte mais beneficiada em relação à largura de banda obtida é a Fonte 2 porque é a que apresenta o maior MCR_i. Quanto às soluções que não dependem dos valores de PCR para o cálculo dos pesos, a terceira proposta (**3 a)** e **3 b)**) para a primeira variante (**V1**) apresenta os mesmos valores, a quarta solução apresenta um valor ligeiramente diferente porque está a considerar o valor realmente disponível para a distribuição pelos VCs ABR, a quinta solução limitou-se a usar o valor total do MCR que não vai variar durante o tempo de vida dos VCs e a sexta solução apresenta uma distribuição dos valores menos agressiva do que a solução **5 V1**.

Tabela 7.2: Resultados numéricos obtidos com a aplicação dos vários critérios.

Critério	**Variante**	**Fonte**	MCR_i $(Mbit/s)$	PCR_i $(Mbit/s)$	Δ_i	ϕ_i	w_i	p_i	$B \times p_i$ $(Mbit/s)$	B_i $(Mbit/s)$
1	V1	**1**	5.00	15.00	10.00	0.67	1.50	0.31	4.66	9.66
		2	20.00	50.00	30.00	0.60	1.67	0.34	5.17	25.17
		3	10.00	25.00	15.00	0.60	1.67	0.34	5.17	15.17
	V2	**1**	–	–	–	–	0.33	0.29	4.41	9.41
		2	–	–	–	–	0.40	0.35	5.29	25.29
		3	–	–	–	–	0.40	0.35	5.29	15.29
2	V1	**1**	5.00	15.00	10.00	0.14	0.41	0.28	4.14	11.14
		2	20.00	50.00	30.00	0.57	0.23	0.40	5.99	23.41
		3	10.00	25.00	15.00	0.29	0.36	0.32	4.87	15.45
	V2	**1**	–	–	–	–	0.41	0.28	4.14	9.14
		2	–	–	–	–	0.23	0.40	5.99	25.99
		3	–	–	–	–	0.36	0.32	4.87	14.87
3 a)	V1	**1**	5.00	15.00	10.00	0.97	1.03	0.32	4.76	9.76
		2	20.00	50.00	30.00	0.87	1.15	0.35	5.31	25.31
		3	10.00	25.00	15.00	0.93	1.07	0.33	4.93	14.93
	V2	**1**	–	–	–	–	0.03	0.14	2.14	7.14
		2	–	–	–	–	0.13	0.57	8.57	28.57
		3	–	–	–	–	0.07	0.29	4.29	14.29
3 b)	V1	**1**	5.00	15.00	10.00	1.26	0.79	0.32	4.76	9.76
		2	20.00	50.00	30.00	1.13	0.88	0.35	5.31	25.31
		3	10.00	25.00	15.00	1.22	0.82	0.33	4.93	14.93
4	V1	**1**	5.00	15.00	10.00	3.00	0.33	0.28	4.14	9.14
		2	20.00	50.00	30.00	2.00	0.50	0.41	6.21	26.21
		3	10.00	25.00	15.00	2.67	0.38	0.31	4.66	14.66
5	V1	**1**	5.00	15.00	10.00	0.86	1.17	0.24	3.57	8.57
		2	20.00	50.00	30.00	0.43	2.33	0.48	7.14	27.14
		3	10.00	25.00	15.00	0.71	1.40	0.29	4.29	14.29
	V2	**1**	–	–	–	–	0.14	0.14	2.14	7.14
		2	–	–	–	–	0.57	0.57	8.57	28.57
		3	–	–	–	–	0.29	0.29	4.29	14.29
6	V1*	**1**	5.00	15.00	10.00	–	0.48	0.24	3.57	8.57
		2	20.00	50.00	30.00	–	0.90	0.45	6.79	26.79
		3	10.00	25.00	15.00	–	0.62	0.31	4.64	14.64

Variante: V1 = $\frac{1}{\phi_i}$, V2 = $1 - \phi_i$

* $k = \frac{1}{N}$

Como conclusão é possível observar que a solução **6 V1** está de acordo com os pressupostos definidos inicialmente e por isso a escolha desta solução apresenta-se como a mais indicada.

7.3.3 Conclusões

Com base na discussão relativa às propriedades de cada solução e aos resultados obtidos com o exemplo numérico apresentado, não existe qualquer dúvida na escolha da solução como critério para o cálculo dos pesos, mas antes de apresentar a solução escolhida são referidas algumas das vantagens das soluções excluídas.

A primeira solução é simples de implementar e para $\forall_i \; MCR_i = 0$ permite configurações com $MCR_i = 0$ e $MCR_i \neq 0$, enquanto que a segunda apresenta uma vantagem adicional que é permitir a escolha de qual dos elementos do peso final será

mais preponderante, o que permite uma infinidade de combinações. A terceira solução "liberta-se" da dependência dos valores de PCR, mas entra em consideração com o valor total da ligação na variante **V1** e com a largura de banda remanescente em relação ao AR quando o que tem de ser distribuído é o valor disponibilizado à categoria de serviço ABR como é o caso da quarta proposta.

Relativamente ao número de cálculos necessários efectuar e à quantidade de informação que é necessário armazenar para o cálculo dos pesos, a segunda solução é a que apresenta o pior resultado e as duas últimas (quinta e sexta) o melhor, porque o valor total de MCR não varia até à entrada ou saída de um VC; quando é estabelecido um VC, o valor do seu MCR_i é distribuído por todos os comutadores, logo também não é necessário estar sempre a calcular. A quarta solução apresenta um bom compromisso entre a simplicidade, o número de instruções a executar e a informação que necessita de ser armazenada nos comutadores para o cálculo dos pesos, mas o seu termo de referência é variável devido à dependência da largura de banda disponível para a categoria de serviço ABR, em cada instante.

Como conclusão poder-se-ia considerar, para além do critério **6 V1**, os critérios **3 b) V1** e **5 V1**, porque ambos permitem configurações constituídas por fontes com $MCR_i = 0$ e fontes com $MCR_i \neq 0$. A única situação que diferencia as duas (**3 b) V1** e **5 V1**) em termos práticos é o termo de referência. Enquanto que a solução **3 b) V1** usa a largura de banda remanescente em relação à capacidade total da ligação, a solução **5 V1** usa um valor que depende de si própria, isto é, "olha" apenas para as necessidades contratualizadas com a rede. Assim, o critério escolhido para aplicar nos EDs na distribuição da largura de banda é o **6 V1**, devido às suas propriedades e aos resultados numéricos alcançados.

7.4 Distribuição da largura de banda remanescente

Com base nos pressupostos discutidos, no critério (7.6) e na expressão do peso definido em (7.62), a largura de banda distribuída a cada VC (assumindo que todas as fontes estão controladas) é dada por

$$B_i = MCR_i + (B - M) \times p_i. \tag{7.63}$$

Assumindo que A representa o débito disponível, N representa o número de fontes a disputar a largura de banda, MCR_i representa o débito mínimo garantido para a fonte i, PCR_i representa o débito máximo desejado para a fonte i e p_i representa o peso atribuído à fonte i na distribuição da largura de banda remanescente, pretende-se saber qual o débito a distribuir por cada uma das fontes em competição.

Sabendo que qualquer algoritmo de distribuição da largura de banda atribuída à categoria ABR deve obedecer à condição

$$\sum_{i=1}^{N} MCR_i \leq A, \tag{7.64}$$

então a largura de banda remanescente que deve ser distribuída equitativamente (no sentido do *max-min fair share*) pelas fontes é

$$B = A - \sum_{i=1}^{N} MCR_i. \tag{7.65}$$

Como o objectivo é calcular um vector de valores $r_i = \{r_1, r_2, ..., r_n\}$ que satisfaça a condição

$$\sum_{i=1}^{N} \delta_i = B \tag{7.66}$$

então $r_i = MCR_i + \delta_i$ representa o débito a distribuir à fonte i desde que

$$r_i \leq PCR_i \tag{7.67}$$

e

$$\sum_{i=1}^{N} r_i \leq A. \tag{7.68}$$

Com base nas condições apresentadas é possível estabelecer a seguinte definição, para a distribuição equitativamente pesada da largura de banda pelos VCs que têm de partilhar o recurso dentro do ED.

Definição 1 *Distribuição equitativamente pesada: Dado um vector com os débitos mínimos garantidos* $\{MCR_1, MCR_2, ..., MCR_N\}$*, o débito máximo disponível, um vector com os débitos máximos desejados* $\{PCR_1, PCR_2, ..., PCR_N\}$ *e calculando dinamicamente (no sentido de alteração do número de fontes), através da expressão (7.62), o vector*

Tabela 7.3: Exemplo de distribuição da largura de banda.

Fonte	MCR_i	PCR_i	Largura de banda ($Mbit/s$)					
			40	75	130		150	
			1^o	1^o	1^o	2^o	1^o	2^o
1	5	45	6.19	14.52	27.62	33.00	32.38	40.00
2	20	150	22.26	38.10	62.98	72.00	72.02	85.00
3	10	25	11.55	22.38	**39.40**	25.00	**45.60**	25.00

de pesos $\{p_1, p_2, ..., p_N\}$*, então o vector de débitos* $\{r_1, r_2, ..., r_n\}$ *que obedece ao critério max-min fairness definido é igual ao vector de débitos* $\{B_1, B_2, ..., B_n\}$ *obtido através da aplicação da expressão (7.63) desde que obedeça às condições impostas por (7.67) e (7.68).*

7.4.1 Exemplo de distribuição da largura de banda

Para exemplificar a distribuição da largura de banda pelos VCs em competição foi utilizado um cenário constituído por fontes com $MCR_i = \{5, 20, 10\}$ e $PCR_i = \{45, 150, 25\}$, com diferentes quantidades de largura de banda disponível $B = \{40, 75, 130, 150\}$ $Mbit/s$, de forma a que fosse possível perceber qual a reacção do critério, quando tem banda disponível que não permite que nenhuma das fontes seja satisfeita e outras que possibilitem ter fontes totalmente satisfeitas e fontes controladas.

Como pode ser observado na Tabela 7.3, para as duas primeiras situações (40 e 75 $Mbit/s$), a largura de banda foi distribuída sem qualquer necessidade de recalcular os pesos e a respectiva distribuição, dado que o total disponível não era suficiente para satisfazer nenhumas das fontes na totalidade. Relativamente às últimas duas quantidades o mesmo já não se passou e foi necessária uma segunda redistribuição para o terceiro (130 $Mbit/s$) e para o quarto caso (150 $Mbit/s$).

7.4.2 Solução algorítmica

Com base na Definição 1 e nos pressupostos anteriores é possível definir um algoritmo que, de forma simples e eficiente realiza uma distribuição equitativa da largura de banda pelas conexões em competição.

Esta solução está centrada no cálculo do débito disponível para a categoria de serviço ABR, em intervalos (de tempo ou no número de células observadas) previamente definidos e na recepção das células BRM. A partir do instante que se obtém o débito

disponível e é recebida uma célula BRM realiza-se a distribuição da largura de banda disponível, com base nos pesos calculados para cada um dos VCs. Após a distribuição é realizado um teste para verificar se o débito obtido para o VC ao qual a célula BRM pertence é maior ou menor do que o débito transportado. Se o débito calculado for superior então é considerado o débito que vem de jusante, se não é usado o débito calculado.

A ideia do algoritmo baseia-se numa distribuição *sucessiva* da largura de banda remanescente pelos VCs em competição. A distribuição *sucessiva* tenta reflectir na prática o conceito teórico em que se baseia o critério *max-min fairness*, ou seja, é realizada uma primeira distribuição da largura de banda e verificado se alguns dos VCs recebeu mais do que o seu PCR; se o resultado for positivo, então a largura de banda atribuída a esse ou esses VCs é o seu PCR sendo a restante redistribuída pelos outros VCs até que não exista nenhum com largura de banda atribuída acima do seu PCR.

Além desta situação é ainda considerada uma questão importante relativamente à distribuição da largura de banda remanescente, porque podem existir casos em que o débito calculado (ERQ_i) seja inferior ao débito de pico da fonte e onde tudo estaria correcto, mas é necessário verificar se o valor obtido não excedeu o débito indicado na célula RM de jusante ($RM.ER$). Se esta situação não fosse contemplada e ocorresse, então existiria uma fracção da largura de banda que não era considerada, e desperdiçada ($RM.ER - ERQ_i$) porque o débito disponibilizado seria sempre o mínimo. Para resolver esta questão foi necessário introduzir uma condição que garantisse o menor número possível de iterações até alcançar a distribuição correcta. Desta forma, este tipo de teste (comparar o débito calculado com o PCR e o $RM.ER$) é realizado apenas quando o VC que está a ser tratado é o mesmo ao qual a célula RM pertence. Contudo, se existir uma configuração em que os VCs estejam sempre congestionados, o algoritmo realiza a distribuição numa única iteração.

Com base na Definição 1 e no critério *max-min fairness* é possível definir um algoritmo para a distribuição equitativa da largura de banda, mesmo quando os valores de MCR e PCR de algumas fontes estão próximos.

/* Inicialização */

N	Número de filas a servir
M	Tamanho do intervalo de medida
TR	*Target Rate* (débito máximo disponível para ABR)
MCR_i	Débito mínimo garantido da fonte i
PCR_i	Débito máximo desejado pela fonte i
MCR	Total dos débitos mínimos garantidos
ERQ_i	Débito atribuído à fonte i
p_i	Peso atribuído à fonte i
k	Grau de congestionamento dentro do comutador
$countCell$	Número de células ABR
CBR	Débito atribuído às conexões CBR
VBR	Débito atribuído às conexões VBR
$inputRateABR$	Débito ABR medido na entrada do ED
$rateABR$	Débito disponível para a categoria de serviço ABR
$rate$	Variável auxiliar para o controlo do débito disponível, em cada momento
$count$	Variável para o controlo do número de iteracções necessárias à distribuição da largura de banda
$distExcesso_i$	Variável de controlo relativa à largura de banda atribuída em execesso à fonte i
AR	Débito total da ligação

01. $TR \leftarrow Link\ Rate$
02. $do\ for\ i = 1\ to\ N$
03. $\quad ERQ_i \leftarrow TR$
04. $countCell \leftarrow 0$
05. $k \leftarrow 0.0$
06. $t_last \leftarrow time()$

/* Recepção de uma célula RM de montante */

07. $MCR_i \leftarrow RM.MCR$
08. $PCR_i \leftarrow RM.PCR$

/* No fim do intervalo de medida */

09. $While\ (On_line)$
10. $\quad if\ (cells\ enter\ queue)$
11. $\quad\quad countCell++$
12. $\quad\quad if\ (countCell\%M == 0)$
13. $\quad\quad\quad inputRateAbr \leftarrow (M \times 424)/(\leftarrow time() - t_last)$
14. $\quad\quad\quad rateABR \leftarrow TR - (CBR + VBR)$
15. $\quad\quad\quad t_last \leftarrow time()$

/* Na recepção de uma célula RM de jusante pertencente ao VC j */

16. $rate \leftarrow rateABR$
17. $do\ for\ i = 1\ to\ N$
18. $\quad distExcesso_i \leftarrow 0$

19. do
20. $MCR \leftarrow 0.0$
21. $NN \leftarrow 0$
22. $do\ for\ i = 1\ to\ N$
23. $if\ (distExcess_i == 0)$
24. $NN \leftarrow NN + 1$
25. $MCR \leftarrow MCR + MCR_i$
26. $do\ for\ i = 1\ to\ N$
27. $if\ (distExcess_i == 0)$
28. $p_i \leftarrow 1/NN + MCR_i/MCR$
29. $do\ for\ i = 1\ to\ N$
30. $if\ (distExcess_i == 0)$
31. $ERQ_i \leftarrow MCR_i + (rate - MCR) \times p_i$
32. $count \leftarrow 0$
33. $banda \leftarrow 0.0$
34. $do\ for\ i = 1\ to\ N$
35. $if\ (i <> j)$
36. $if\ (ERQ_i > PCR_i)$
37. $banda \leftarrow banda + PCR_i$
38. $distExcess_i \leftarrow 1$
39. $ERQ_i \leftarrow PCR_i$
40. $else$
41. $count \leftarrow count + 1$
42. $else$
43. $if\ (ERQ_i > PCR_i)$
44. $if\ (ERQ_i > RM.ER\ and\ RM.ER < PCR_i)$
45. $banda \leftarrow banda + RM.ER$
46. $distExcess_i \leftarrow 1$
47. $ERQ_i \leftarrow RM.ER$
48. $else$
49. $banda \leftarrow banda + PCR_i$
50. $distExcess_i \leftarrow 1$
51. $ERQ_i \leftarrow PCR_i$
52. $else$
53. $if\ (ERQ_i > RM.ER)$
54. $banda \leftarrow banda + RM.ER$

55. $distExcess_i \leftarrow 1$
56. $ERQ_i \leftarrow RM.ER$
57. $else$
58. $count \leftarrow count + 1$
59. $rate \leftarrow rate - banda$
60. $while\ (count < N)$
61. $ERQ_j \leftarrow min(ERQ_j, rateABR)$
62. $RM.ER \leftarrow ERQ_j$

7.5 Resultados de simulação

Nesta secção é feita a análise de desempenho com base nos resultados obtidos com o algoritmo de distribuição da largura de banda. Estes resultados foram obtidos numa grande variedade de configurações, incluindo configurações com e sem fontes VBR, fontes ABR e VBR a iniciarem a sua actividade decaladas no tempo, etc. No entanto, só serão apresentados os resultados obtidos em duas configurações que permitem observar a maioria das situações que podem levar à ocorrência de problemas.

Com estes resultados é possível analisar a eficiência, *fairness*, a escalabilidade, a estabilidade, a ocupação das filas de saída, a eficiência na utilização dos recursos e a adaptação à variação da largura de banda para fontes com vários modelos de tráfego.

Os resultados serão apresentados na forma de dois gráficos: um gráfico composto pelo débito permitido (ACR_i) a cada uma das fontes e a ocupação das linhas ($Link_i$) que interligam as fontes e os comutadores e um segundo gráfico referente à ocupação das filas em cada porta de saída de cada comutador.

7.5.1 Configuração simples com fontes com diferentes débitos de pico

O objectivo desta subsecção é estudar o comportamento do algoritmo de controlo de fluxo quando a configuração utilizada é constituída por uma fonte VBR e por fontes ABR que estão sempre controladas (encontram uma ligação congestionada), por fontes ABR que antes da fonte VBR iniciar a transmissão do seu *burst* não se encontram controladas (isto é, transmitem com o seu débito de pico), mas após a sua entrada em actividade

Tabela 7.4: Resultados esperados nos testes de simulação com uma configuração com cinco fontes ABR (com diferentes débitos de pico) e uma VBR.

Fonte	**MCR** (*Mbit/s*)	**PCR** (*Mbit/s*)	Largura da Banda (*Mbit/s*)							
			VBR *off*				VBR *on*			
			1^a		2^a		1^a		2^a	
			$B = 150$		$B = 129$		$B = 50$		$B = 44$	
			p_i	LB	p_i	LB	p_i	LB	p_i	LB
A1	0.5	3	0.2	30.00	–	3.00	0.2	10.00	–	3.00
A2	0.5	3	0.2	30.00	–	3.00	0.2	10.00	–	3.00
A3	0.5	20	0.2	30.00	–	20.00	0.2	10.00	0.33	14.67
A4	0.5	15	0.2	30.00	–	15.00	0.2	10.00	0.33	14.67
A5	0.5	150	0.2	30.00	1.0	109.00	0.2	10.00	0.33	14.67

passam a ser controladas e por fontes ABR que nunca estão controladas (transmitem sempre ao seu débito de pico).

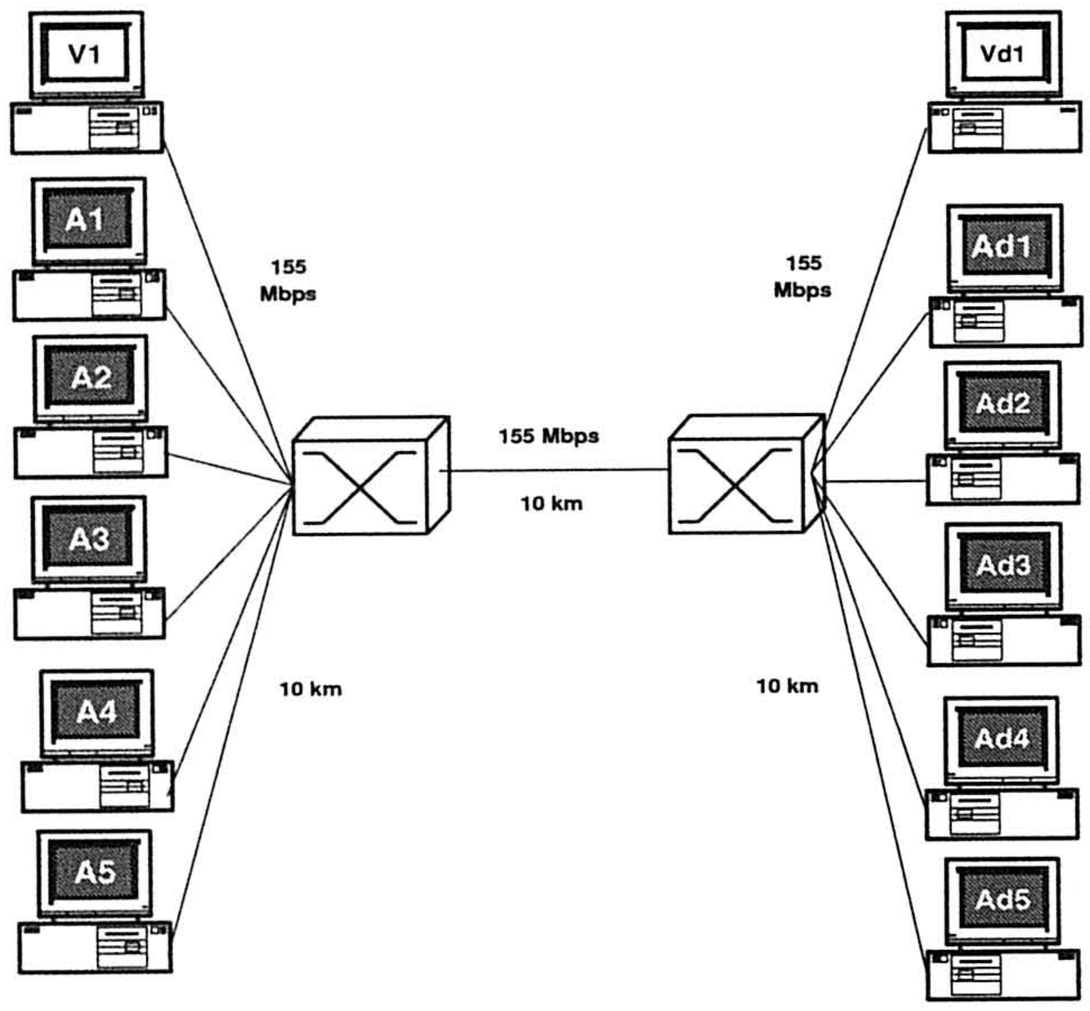

Figura 7.3: Configuração com fontes ABR ponto-a-ponto.

Os testes realizados tiveram como base a configuração da Figura 7.3. A primeira fonte é VBR e as fontes A1 e A2 são ABR com um débito de pico de 3 $Mbit/s$, a fonte A3, também ABR, tem um débito de pico de 20 $Mbit/s$ e as fontes A4 e A5 com débitos de pico de 15 e 150 $Mbit/s$, respectivamente. A distribuição da largura de banda pelos diversos VCs será diferente, dependendo do período de actividade ou repouso da fonte VBR; essa distribuição está representada na Tabela 7.4.

Os critérios de análise de desempenho foram todos alcançados, como pode ser observado na Figura 7.4.

Figura 7.4: Resultados obtidos com a configuração com fontes com diferentes débitos de pico.

7.5.2 Configuração complexa com fontes sempre controladas

O principal objectivo a verificar com uma configuração deste tipo é o comportamento do algoritmo numa configuração complexa, incluindo fontes ABR e VBR a operarem em conjunto, mas com as distâncias das fontes ABR diferentes, isto é, com VCs ABR que apenas atravessam uma única linha principal (*Single Hop*, SH) e VCs que atravessam mais do que uma linha de interligação dos comutadores (*Multiple Hop*, MH), facto que levanta uma série de problemas.

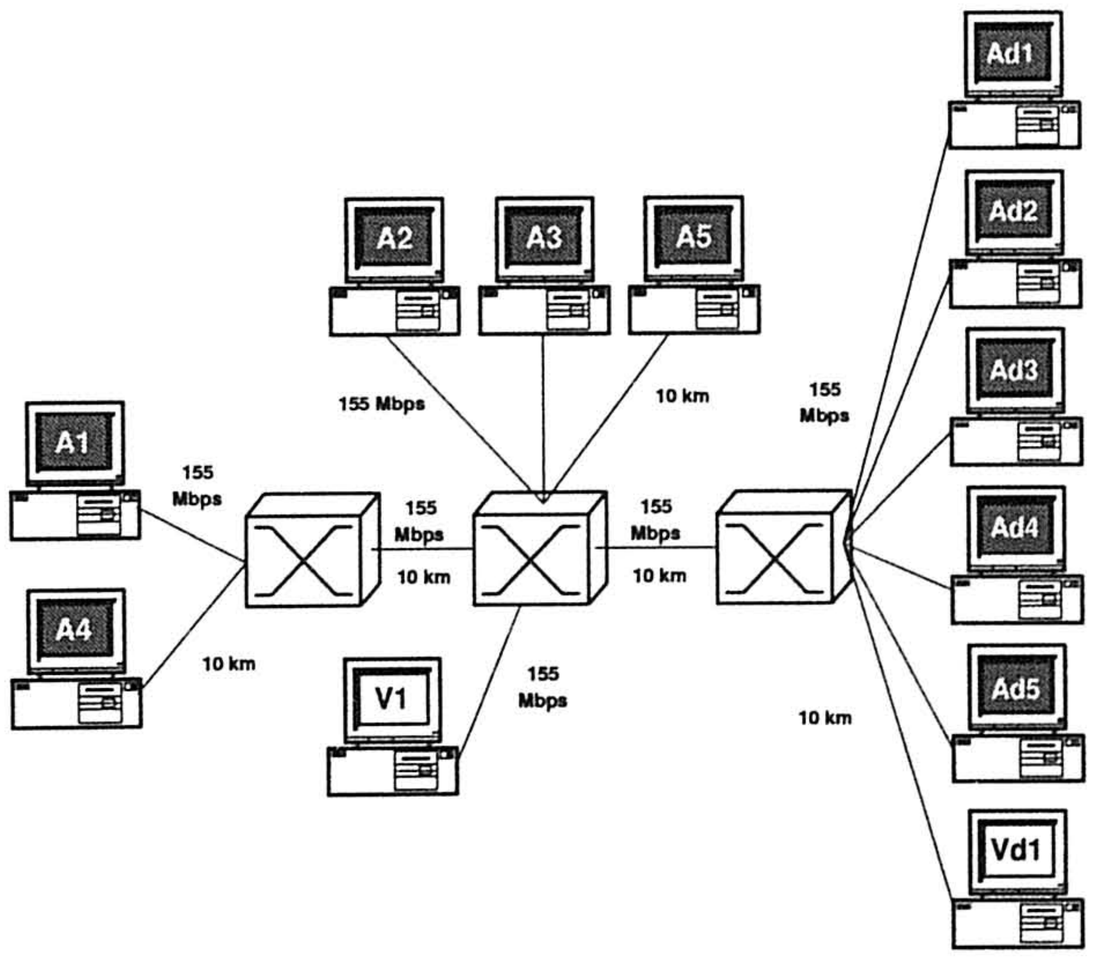

Figura 7.5: Configuração com fontes ABR e VBR ponto-a-ponto.

Os resultados obtidos tiveram como base a configuração da Figura 7.5. Esta

Tabela 7.5: Resultados esperados nos testes de simulação com a segunda configuração com fontes sempre controladas.

Fontes	**MCR** ($Mbit/s$)	**PCR** ($Mbit/s$)	Δ_i	Largura da Banda ($Mbit/s$)			
				VBR off		VBR on	
				1^a		1^a	
				$B = 150$		$B = 100$	
				p_i	LB	p_i	LB
A1	20	150	130	0.23	35.75	0.23	24.50
A2	30	150	120	0.29	50.13	0.29	35.75
A3	5	100	145	0.13	14.19	0.13	7.63
A4	15	150	135	0.19	28.56	0.19	18.88
A5	10	150	140	0.16	21.38	0.16	13.25

configuração é constituída por cinco fontes ABR, sendo os VCs de duas do tipo MH e as restantes, incluindo a fonte VBR (gera *bursts* de 10 ms com um débito de pico de 50 $Mbit/s$) do tipo SH. A distribuição da largura de banda pelos diversos VCs será diferente, dependendo do período de actividade ou repouso da fonte VBR; essa distribuição está na Tabela 7.5.

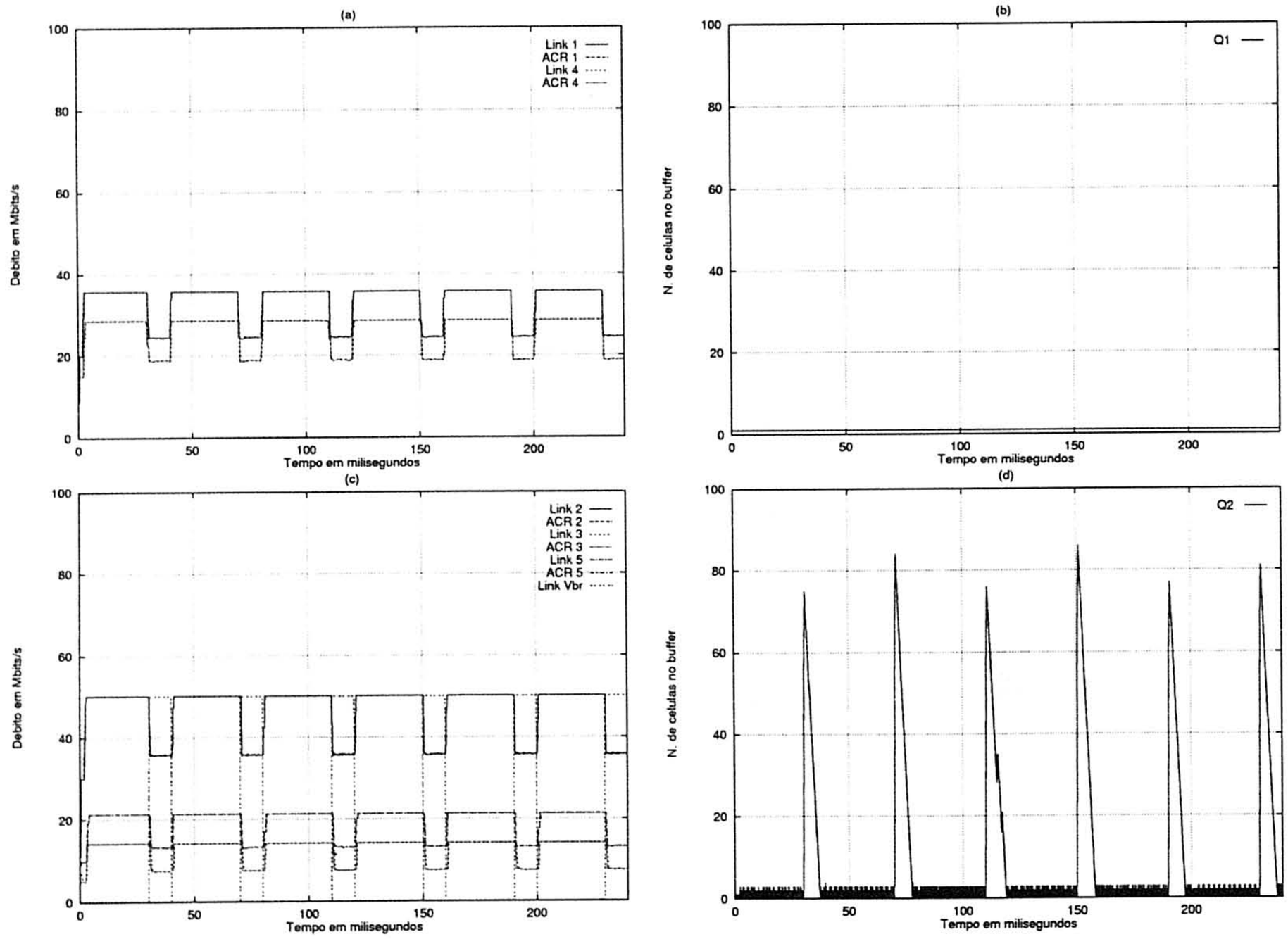

Figura 7.6: Resultados do primeiro e segundo comutador na configuração complexa com fontes sempre controladas.

Os resultados apresentados nos gráficos da Figura 7.6 referem-se ao comporta-

Tabela 7.6: Resultados esperados nos testes de simulação com a segunda configuração com fontes com diferentes débitos de pico.

Fontes	**MCR** ($Mbit/s$)	**PCR** ($Mbit/s$)	Largura da Banda ($Mbit/s$)							
			VBR off				VBR on			
			1^a		2^a		1^a		2^a	
			$B = 150$		$B = 95$		$B = 100$		$B = 85$	
			p_i	LB	p_i	LB	p_i	LB	p_i	LB
A1	10	15	0.18	26.50	–	15.00	0.18	17.33	–	15.00
A2	20	40	0.27	44.00	–	40.00	0.27	30.67	0.33	31.38
A3	5	100	0.14	17.75	0.25	21.25	0.14	10.67	0.18	11.13
A4	15	150	0.23	35.25	0.42	42.08	0.23	24.00	0.28	24.63
A5	10	150	0.18	26.50	0.33	31.67	0.18	17.33	0.23	17.88

mento dos dois comutadores. No primeiro comutador verifica-se a influência da fonte VBR nos seus momentos de transmissão, nomeadamente no gráfico (a) onde se verifica um decréscimo no débito atribuído às fontes A1 e A4; os valores obtidos correspondem aos valores da Tabela 7.5, o que quer dizer que o mecanismo converge rápida e correctamente para os valores ideais. Nos instantes em que a fonte VBR termina o seu *burst*, a largura de banda é novamente redistribuída, sendo os novos débitos obtidos os esperados e o mesmo se aplica ao tempo de convergência. Os resultados obtidos no segundo comutador mostram que o comportamento observado é bom, uma vez que a distribuição da largura de banda durante os períodos *On* e *Off* da fonte VBR são sempre os calculados inicialmente a par dos tempos de convergência que são muito baixos, o que implica um bom desempenho (ver gráficos (c) e (d)).

7.5.3 Configuração complexa com fontes com diferentes débitos de pico

Pretende-se igualmente verificar se os objectivos identificados são também alcançados com este algoritmo e com a mesma configuração (ver Figura 7.5), mas com fontes com diferentes débitos de pico e com a fonte VBR a gerar *bursts* de 20 ms.

Os resultados apresentados nos gráficos da Figura 7.7 referem-se ao comportamento dos dois comutadores. No primeiro, a existência de uma fonte controlada e uma outra a transmitir sempre ao seu débito de pico, em ambos os períodos (actividade e repouso) da fonte VBR, como pode ser observado nos gráficos (a) e (b), os valores obtidos correspondem aos valores da Tabela 7.6, o que quer dizer que o mecanismo converge rápida e correctamente para os valores esperados. Os valores obtidos para o segundo comutador apresentam o comportamento esperado, uma vez que a distribuição da lar-

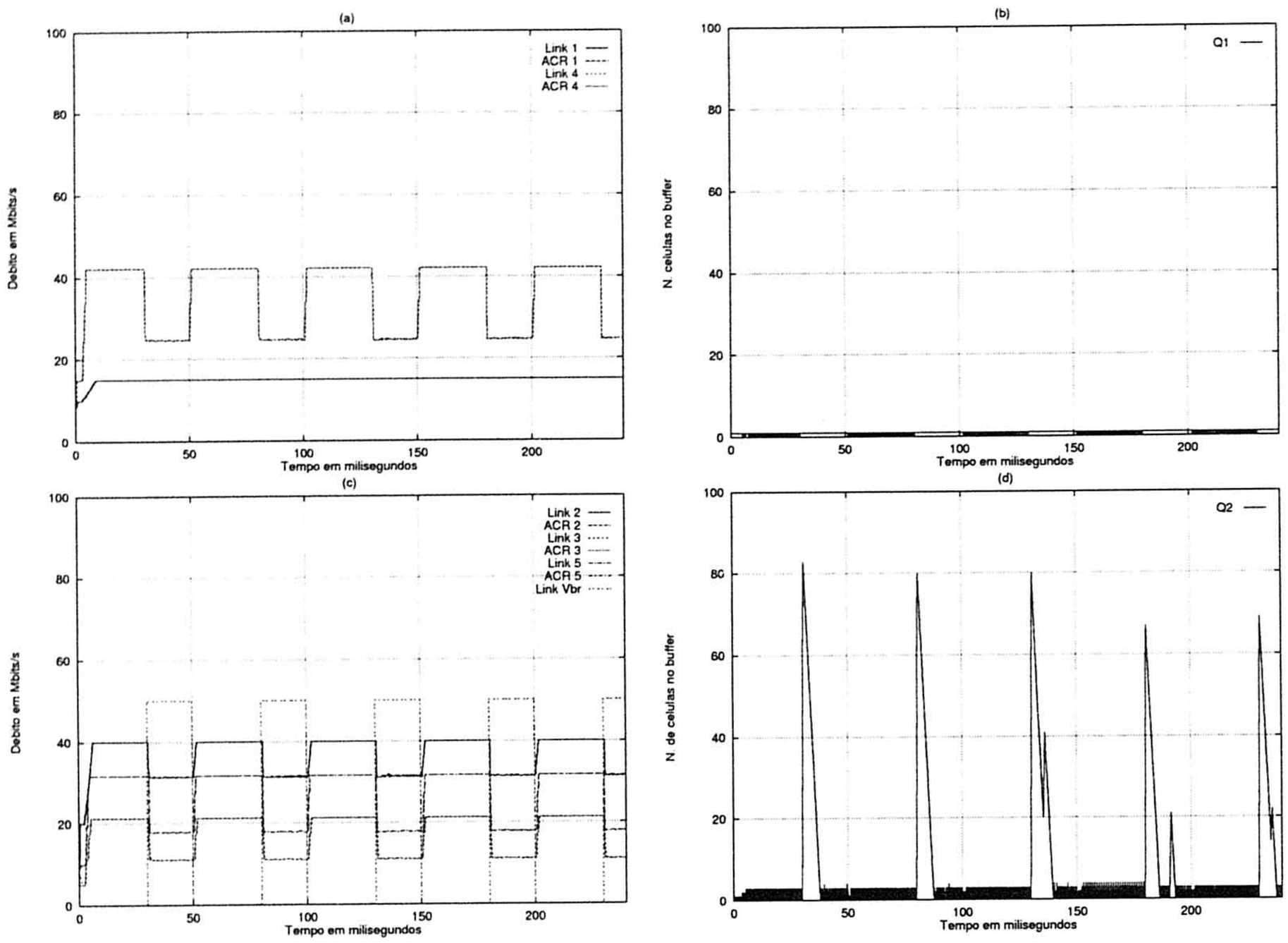

Figura 7.7: Resultados do primeiro e segundo comutador na configuração complexa com fontes com diferentes débitos de pico.

gura de banda, durante os períodos *On* e *Off* da fonte VBR, são sempre os calculados inicialmente, a par de tempos de convergência baixos (gráfico (c)) e uma ocupação do *buffer* sempre controlada (gráfico (d)), o que levou à obtenção de um bom desempenho global.

7.6 Conclusões

Nesta secção foram abordados assuntos relacionados com a distribuição da largura de banda para a categoria de serviço ABR. A ideia de propor soluções para a distribuição da largura de banda dentro dos EDs teve por base a necessidade de atribuir este recurso ao VC que agrega os fluxos IP. Para a resolução deste problema era possível utilizar uma das muitas propostas existentes na literatura, mas estas apresentam vários graus de complexidade e diferentes técnicas para a distribuição da largura de banda, para além de não considerarem a questão relacionada com a possibilidade de ter fontes a transmitir ao seu débito de pico.

A solução proposta centrou-se, por um lado, na resolução dos problemas encontrados na literatura e, por outro lado, na introdução de um critério para o cálculo dos pesos a atribuir a cada VC na distribuição da largura de banda remanescente. Os resultados obtidos foram os previstos pelos exemplos numéricos apresentados, atingindo-se os vários objectivos que se estabelecem para este tipo de algoritmos, nomeadamente *fairness*, a escalabilidade, a estabilidade, o controlo da ocupação das filas de saída, a eficiência na utilização dos recursos e a adaptação à variação da largura de banda.

Capítulo 8

Consolidação de células de gestão de recursos para conexões ponto-a-multiponto

A ideia de apresentar uma solução integrada só faz sentido se a intervenção não se limitar aos EDs, mas estender-se também à rede de núcleo. Nesse sentido, e como a categoria de serviço escolhida foi a ABR, serão discutidos aspectos relacionados com as conexões do tipo ponto-a-multiponto. Após a referida discussão será apresentada uma proposta para a resolução dos problemas típicos deste tipo de conexões multiponto.

Na primeira secção é realizado um breve enquadramento dos assuntos relacionados com as conexões multiponto. Na segunda secção são discutidos os aspectos a ter em consideração no desenvolvimento de esquemas para conexões ponto-a-multiponto. Na terceira secção é apresentada uma proposta para a consolidação das células de gestão de recursos para conexões ponto-a-multiponto. A solução algorítmica é objecto da quarta secção. Na quinta secção é realizada a respectiva avaliação, através de resultados obtidos por simulação, com base em configurações de referência. Na última secção são apresentadas as respectivas conclusões.

8.1 Introdução

A necessidade crescente de comunicação entre membros de uma comunidade, ou mesmo entre comunidades, levou ao aparecimento de aplicações e protocolos capazes de atingir

esse propósito. Foi com vista à satisfação destas necessidades que apareceram os protocolos para o suporte de aplicações do tipo multiponto, que permitem que mais do que dois utilizadores troquem informação entre si.

Qualquer uma destas aplicações usa um protocolo de transporte para satisfazer o suporte a comunicações *multicast* (proposta inicial realizada por Deering [29]). A aposta nesta estrutura de suporte às aplicações deve-se ao facto de que outras tecnologias não abrangem ainda o espaço suficiente (a estrutura da Internet, por exemplo) para que os protocolos possam tirar partido de outras características (por exemplo, QoS). No entanto, a nova geração de aplicações, em especial as interactivas, introduzem novos requisitos na comunicação e neste caso uma das tecnologias que vai ao seu encontro é a tecnologia ATM como discutido em [68].

Para este tipo de aplicações são necessários diferentes tipos de conexões, de acordo com o fim a que se destinam, por exemplo, ponto-a-multiponto para aplicações de distribuição de *software*, multiponto-a-ponto para aplicações de sistemas de controlo e multiponto-a-multiponto para videoconferência. Qualquer um dos três tipos de conexões tem características próprias e levanta problemas específicos, podendo destacar-se o *feedback implosion* e o *consolidation noise* nas conexões ponto-a-multiponto, o *VC-merge* (faz aparecer o problema de *cell interleaving* quando usado AAL5) nas conexões multiponto-a-ponto e, por último, para as conexões multiponto-a-multiponto a conjugação dos três problemas.

Para apresentar uma solução integrada é necessário fazer uma intervenção no ED e na rede de núcleo. Relativamente à intervenção na rede de núcleo, a escolha de VCs ponto-a-ponto não levanta problemas de maior porque existem já soluções com um elevado grau de sofisticação (algumas com definições formais [23] ou [7], por exemplo), enquanto que a utilização de VCs do tipo ponto-a-multiponto levantam problemas adicionais (problemas já caracterizados na secção 2.4.1 do Capítulo 2).

Neste capítulo será apresentada uma proposta referente à consolidação de células de gestão e ao controlo de fluxo e de congestionamento em conexões multiponto, nomeadamente ponto-a-multiponto. Apesar da existência de um conjunto considerável de trabalhos nesta área é necessário salientar que ainda não existe um *standard* definido pelo ATM Forum, ou por outro qualquer organismo de normalização, relativamente às conexões multiponto em redes ATM. Antes de apresentar qualquer solução impõe-se ob-

servar um conjunto de aspectos a ter em consideração no desenvolvimento dos esquemas para conexões ponto-a-multiponto da categoria de serviço ABR, bem como discutir os resultados obtidos através de simulação sobre configurações de referência.

8.2 Aspectos a ter em consideração no desenvolvimento de esquemas para conexões ponto-a-multiponto

As dificuldades colocadas a este tipo de mecanismos foram discutidas na secção 2.4.1 do Capítulo 2 e podem-se resumir como o compromisso entre o *receber a informação correcta ou receber a informação parcialmente correcta*, mas conseguir uma adaptação e uma reação, o mais rapidamente possível ao estado do sistema. Este e outros compromissos devem ser considerados no desenvolvimento de esquemas para este tipo de conexões.

Para além do compromisso referido é também necessário observar outras características tais como, a complexidade, a escalabilidade, o *overhead*, o tempo de resposta às perturbações e a solução para o *consolidation noise*. Alguns autores [36, 34] fazem um conjunto de considerações relativamente a este tipo de compromissos que se podem resumir da seguinte forma:

- **Qual o componente que deve gerar as células BRM ?** Se a opção do esquema passar por ser o ponto de agregação a realizar essa operação, então a complexidade do comutador vai aumentar.

- **Deverá ou não o ponto de agregação esperar pela recepção de uma célula BRM, de cada um dos ramos a jusante, antes de passar a informação para montante ?** Como observado anteriormente, a tomada de decisão relativamente a esta questão tem essencialmente a ver com a política mais ou menos conservadora do mecanismo; no entanto, se a opção for esperar por todas as células, pode levar ao aparecimento de outros problemas tais como aumentar a complexidade e o tempo de resposta do esquema relativamente à reacção ao problema do congestionamento.

- **Qual a forma de controlar o número de células FRM geradas e o número de células BRM tratadas pela fonte ?** É outra das questões que não tem uma

resposta única porque depende do tipo de esquema que se quer implementar; por exemplo, se a opção for esperar por todas as células BRM dos ramos de jusante conectados a um comutador e só depois enviar uma célula BRM para montante com o resultado dessa consolidação, assim o número de células geradas (FRM) e tratadas (BRM) é igual; no entanto, o esquema permite o aparecimento do problema discutido no ponto anterior.

- **Como deve operar o comutador quando é em simultâneo um ponto de agregação e um ponto de enfileiramento ?** A questão apresentada é importante devido à necessidade de perceber quando é que o algoritmo que calcula o débito deve ser executado, isto é, se deve ser executado por cada célula BRM recebida ou só quando é enviada para montante; é mais uma questão específica da implementação.

- **Quais as implicações dos esquemas que retornam uma célula BRM por cada célula FRM recebida ?** A questão coloca-se devido ao problema da escalabilidade, porque um esquema que opere dessa forma irá ser dependente do número de níveis da árvore *multicast* que é uma situação indesejável relativamente à possibilidade do seu crescimento.

- **Como devem ser armazenados os valores das variáveis e *flags* utilizadas na operação do esquema ?** A forma de armazenamento dos valores das variáveis e *flags* não pode ser realizado da mesma forma que nos esquemas ponto-a-ponto, mas por VCs nas conexões ponto-a-multiponto. Estes valores são comuns a todos os ramos pertencentes a uma mesma sessão dentro de um comutador.

- **Como devem ser tratados os ramos que não respondem ?** Se o esquema utilizado é conservativo, isto é, espera pelo *feedback* de todos os ramos de jusante antes de enviar a informação consolidada para montante, então deve usar-se um relógio por cada ramo, e assim que o tempo se esgotar sem receber qualquer informação de *feedback*, esse ramo deve ser considerado fora da sessão, e só depois do tempo terminado é que o comutador irá fornecer para montante a informação consolidada.

Como observado ao longo da apresentação das soluções descritas na literatura

(Capítulo 4), existiu uma grande dificuldade em conseguir responder a todas as questões numa única solução, devido ao tipo de compromissos que é necessário estabelecer relativamente à questão a que se quer conferir maior importância. Da mesma forma que as soluções propostas na literatura tiveram estas preocupações, a solução apresentada nesta secção conseguiu de algum modo conciliar estas questões e dar resposta a todas numa única solução algorítmica, com excepção do problema apontado na última questão (Como devem ser tratados os ramos que não respondem ?), porque assumiu-se que todos os ramos respondiam sempre.

8.3 Definição do modelo

A solução apresentada para este tipo de conexões tem em atenção os vários algoritmos propostos na literatura e em especial o apresentado em [36]. Com base nos problemas encontrados nestes algoritmos é possível propor uma nova solução para atingir quatro objectivos. Como primeiro objectivo pretende-se dar uma resposta rápida e precisa ao congestionamento para montante (por exemplo, no algoritmo proposto em [36], esta operação é referida apenas através de uma forma alternativa, ver secção 4.3.2). Em segundo lugar, o número de células BRM geradas não deve depender directamente do algoritmo de distribuição de largura de banda, isto é, qualquer que seja o algoritmo de distribuição da largura de banda deve ser independente do número de células BRM geradas (por exemplo na proposta [36] é dependente), de forma a que o algoritmo de consolidação de células de gestão de recursos possa interactuar com qualquer solução. Como terceiro objectivo, pretende-se utilizar os resultados produzidos pelo algoritmo de distribuição da largura de banda no algoritmo de consolidação e não chamá-lo sempre que uma célula BRM é recebida (obrigando à sua execução e ao consequente aumento do tempo de execução do algoritmo de consolidação dentro dos comutadores), como acontece, por exemplo em [36]. Por último, tenta-se reduzir o mais possível a complexidade de implementação, por exemplo, evitando a complexidade introduzida no esquema apresentado em [19, 20].

Em termos práticos o que se verifica em alguns dos algoritmos publicados [97, 106, 95, 79, 80, 19, 20] é que quando o congestionamento é detectado, os algoritmos têm de esperar por *feedback* de todos os ramos, antes de indicar essa informação para

a fonte, podendo a fonte continuar a transmitir ao mesmo débito ou até aumentá-lo e, consequentemente, agravar o congestionamento.

A ideia base de suporte ao algoritmo é evitar uma resposta lenta às transições quando existe um congestionamento excessivo, isto é, não é necessário esperar por *feedback* de todos os ramos, mas comunicar imediatamente o congestionamento à fonte; no entanto, nos casos em que não existe uma indicação de congestionamento é melhor esperar pelo *feedback* de todos os ramos, para garantir a eliminação do *consolidation noise* e, por consequência, o *feedback implosion*. Contudo, existirão momentos em que o congestionamento se vai verificar e ainda não foi recebido *feedback* de todos os ramos de jusante e, por isso, é necessário apresentar alternativas para fazer reduzir o débito da fonte.

A ideia principal da solução é conceber formas alternativas de comunicar o novo débito à fonte como, por exemplo, para além de transmitir uma célula BRM para montante após ser recebida pelo menos uma célula BRM de cada um dos ramos a jusante, estabelecer condições que permitam enviar uma nova célula BRM sem agravar o *feedback implosion*. Esta característica garante que o débito veiculado para as fontes é o débito mínimo entre todos os ramos e desta forma a solução não sofre do problema de *consolidation noise*. Entretanto, esta solução agravará o congestionamento sempre que este ocorrer antes de ser recebida uma célula BRM de cada um dos ramos. Para resolver este problema, o comutador pode gerar células BRM após a detecção do congestionamento, que transportarão para montante o débito mínimo obtido até ao momento. Esta solução é diferente das apresentadas inicialmente porque foram introduzidas duas características que tornam o algoritmo mais funcional, devido à forma como o comutador informa os comutadores ou as fontes que se encontram a montante. Primeiro, o comutador gera **apenas uma** célula BRM nos momentos em que a ocupação da fila seja maior do que o limiar superior (Q_H) e, em segundo lugar, permite que o comutador transmita para montante **apenas a primeira** célula BRM recebida de jusante a partir do momento que a fila tenha uma ocupação superior ao limiar inferior (Q_L). É de notar que quando uma célula BRM é retornada devido à detecção de congestionamento antes do *feedback* ter sido recebido de todos os ramos, os contadores e os valores dos registos não são reinicializados para que o processo decorra com toda a normalidade.

As duas formas alternativas de enviar para montante o novo débito e a indicação

de congestionamento levantam algumas questões; por exemplo, a partir do instante em que a ocupação da fila de saída ultrapassa o limiar superior, o comutador gera uma célula BRM, o que levaria à geração de células BRM enquanto a ocupação da fila fosse superior ao respectivo limiar. Esta situação não irá acontecer porque o algoritmo de consolidação irá apenas gerar uma e uma só célula BRM, por cada momento em que a ocupação da fila de saída ultrapassa o limiar superior. Relativamente à segunda alternativa para comunicar o estado do comutador e o respectivo débito (também será controlado) é deixada passar para montante apenas uma célula BRM, sempre que o limiar inferior foi ultrapassado e a taxa de crescimento da fila é superior a 1.

Com base na solução apresentada para o primeiro objectivo é fácil de estender a explicação para conseguir atingir o segundo, isto é, o algoritmo de consolidação gera células BRM e envia-as para montante, de acordo com a ocupação da fila de saída, e não com o débito calculado pelo algoritmo de controlo de fluxo e de congestionamento, o que torna o número de células geradas e deixadas passar para montante dependente apenas do algoritmo de consolidação.

O último objectivo é conseguir reduzir a complexidade de implementação e o *overhead* imposto pelo número elevado de instruções. A redução da complexidade é conseguida através da utilização apenas de um *bit* por ramo, para verificar se já foi recebida uma célula BRM de jusante e, para reduzir o *overhead*, utilizar apenas os valores obtidos com o algoritmo de controlo de fluxo e de congestionamento e não chamá-lo em determinados momentos para os calcular.

O algoritmo oferece uma resposta rápida ao congestionamento quando este é detectado, quer quando recebe de jusante essa indicação, quer quando o congestionamento acontece no próprio ponto de consolidação; quando o ponto de consolidação é um comutador, o algoritmo de controlo de fluxo e de congestionamento proposto pelo autor [81, 82] é executado, independentemente de este ser ou não ponto de consolidação. Contudo, quando a célula BRM está para ser escalonada é verificado se o valor mínimo entre todos os débitos recebidos de jusante é inferior ao ERQ calculado pelo respectivo algoritmo de distribuição de largura de banda. Esta situação é, por exemplo, o oposto ao modelo apresentado em [36], em que nos casos em que o ponto de consolidação é um comutador, o algoritmo ERICA [60, 67] é chamado sempre que uma célula BRM é recebida, e não apenas quando uma célula BRM é enviada, o que envolve alguma

complexidade adicional.

8.4 Solução algorítmica

Nesta secção é apresentado o algoritmo de consolidação proposto e implementado nos comutadores que têm um papel de pontos de consolidação [83].

Em resumo, o modelo proposto espera por todas as células BRM se não for detectado congestionamento, envia células BRM em três condições (se todas as células BRM foram recebidas, se existe início de congestionamento ou se está num estado de congestionamento), o controlo do débito é explícito porque utiliza um algoritmo de distribuição da largura de banda e a informação que necessita armazenar ocupa 1 *bit* por ramo, como pode ser observado no pseudocódigo do algoritmo apresentado de seguida.

$GerarBRM$	Variável que indica se já foi gerada alguma célula BRM;
$PassarBRM$	Variável que indica se já foi passada para montante alguma célula BRM;
$NumBRMReceived$	Número de células BRM recebidas de jusante;
$ReceivedBRM_j$	Indica se já foi recebida alguma célula BRM do ramo j;
MER	Contêm o débito mínimo entre todas as célula BRM recebidas dos ramos de jusante;
$TotalBranchNum$	Número total de ramos pertencentes a uma determinada árvore num comutador.

/* Inicialização */

01. $GerarBRM \leftarrow 0$
02. $PassarBRM \leftarrow 0$
03. $NumBRMReceived \leftarrow 0$
04. $\forall_j(ReceivedBRM_j) \leftarrow 0$
05. $MER \leftarrow PCR$
06. $MCI \leftarrow 0$
07. $MNI \leftarrow 0$

/* Após receber uma célula FRM */

08. Transmitir a célula FRM para todos os ramos a jusante.
09. Processar a célula RM como no algoritmo ponto-a- ponto.

/* Após receber uma célula BRM do ramo i */

10. $if\ (ReceivedBRM_i == 0)$

11. $ReceivedBRM_i \leftarrow 1$
12. $NumBRMReceived \leftarrow NumBRMReceived + 1$
13. $MER \leftarrow min(ER, MER)$
14. $MCI \leftarrow or(CI, MCI)$
15. $MNI \leftarrow or(NI, MNI)$
16. $if\ (NumBRMReceived == TotalBranchNum)$
17. $BRM(ER \leftarrow MER, CI \leftarrow MCI, NI \leftarrow MNI)$
18. $\forall_j(ReceivedBRM_j) \leftarrow 0$
19. $MER \leftarrow$ Total ABR rate
20. $MCI \leftarrow 0$
21. $MNI \leftarrow 0$
22. $GerarBRM \leftarrow 0$
23. $PassarBRM \leftarrow 0$
24. Transmitir a célula BRM para montante
25. *else*
26. $if\ (Q(t) > Q_L\ And\ k > 1\ And\ PassarBRM == 0)$
27. $BRM(ER \leftarrow min(MER, ERQ), CI \leftarrow MCI, NI \leftarrow MNI)$
28. Transmitir a célula BRM para montante
29. $PassarBRM \leftarrow 1$
30. *else*
31. $if\ (Q(t) > Q_H\ And\ GerarBRM == 0)$
32. Gerar uma célula BRM
33. $BRM(ER \leftarrow min(MER, ERQ), CI \leftarrow MCI, NI \leftarrow MNI)$
34. Transmitir a célula BRM para montante
35. $GerarBRM \leftarrow 1$
36. *else*
37. Eliminar a célula BRM
38. $GerarBRM \leftarrow 0$
39. $PassarBRM \leftarrow 0$

Este algoritmo usa um registo designado por *Minimum Explicit Rate* (MER) e duas *flags* MCI e MNI para cada VC ponto-a-multiponto. O valor de MER é inicialmente atribuído ao PCR, enquanto que as *flags* MCI e MNI são inicializadas a zero. O registo MER contém o débito mínimo entre todos os ramos a jusante do comutador, enquanto

que MCI é o operador lógico OR da indicação do congestionamento, e MNI é o operador lógico OR da indicação de não incremento do débito de todas as células BRM recebidas de jusante.

Quando o comutador recebe uma célula BRM do ramo i, a variável $ReceivedBRM_i$ é atribuída a 1 se ainda não tiver sido recebida uma célula desse ramo. Nesta situação o contador $NumBRMReceived$ é incrementado de uma unidade. De seguida, o registo MER recebe o resultado da operação $min(MER, ER)$. Assim, o comutador obtém o valor mínimo do débito entre todos os ramos a jusante. Em simultâneo as *flags* MCI e MNI são também actualizadas, de acordo com os valores transportados na célula BRM. Quando o conteúdo da variável $NumBRMReceived$ for igual ao número de ramos ($TotalBranchNum$), a célula BRM recebida é encaminhada para montante. Esta célula será transmitida com o valor do registo MER atribuído ao campo ER e os campos CI e NI com o valor das *flags* MCI e MNI. Após esta operação o comutador reinicializa o valor de MER, a variável $ReceivedBRM_j$, as *flags* MCI e MNI, a variável que indica a geração de uma célula BRM ($GerarBRM$) e a variável que indica a passagem da primeira célula BRM ($PassarBRM$).

Sempre que o contador $NumBRMReceived$ é diferente de $TotalBranchNum$, o algoritmo analisa o tamanho da fila para verificar se é superior a Q_L e se está a crescer ($k > 1$). Nesta situação, um resultado positivo indica que uma situação de congestionamento pode estar a iniciar-se e, antes que se agrave, o comutador encaminha para montante a primeira célula BRM que receber de jusante, com os valores $ER \leftarrow min(MER, ERQ_i)$, $CI \leftarrow MCI$ e $NI \leftarrow MNI$. A passagem apenas da primeira célula BRM após a ocupação ser superior ao limiar inferior e a indicação de crescimento é garantida pela variável $PassarBRM$, se esta tiver atribuído o valor zero. Se o resultado for negativo é verificado se o congestionamento teve início ($Q(t) > Q_H$). Se o resultado for positivo então é gerada uma célula BRM e atribuído o $min(MER, ERQ_i)$, MCI e MNI aos campos da célula (ER, CI e NI), e esta é enviada para montante. Para evitar a geração sucessiva de células BRM enquanto o congestionamento persistir é utilizada uma variável $GerarBRM$ que é atribuída a 1 sempre que se gera uma célula deste tipo. A inexistência deste controlo levaria o algoritmo a produzir um elevado número de células e, consequentemente reaparecia o problema *feedback implosion* que se pretende eliminar.

Tabela 8.1: Parâmetros das fontes e dos comutadores.

Parâmetros	**Valores**
Fontes ABR	
PCR (Peak Cell Rate) (em Mbit/s)	149.76
Nrm (Number of RM Cells) (em células)	32
ICR (Initial Cell Rate) (em Mbit/s)	7.49
MCR (Minimum Cell Rate) (em Mbit/s)	0.149
AIR (Additive Increase Rate Factor)	0.212
RDF (Reduction Factor)	512
ACR (Allowed Cell Rate (em Mbit/s)	Valor dinâmico
Comutador	
Q_H (High Threshold) (em células)	80
Q_L (Low Threshold) (em células)	60
N (Measurement Interval) (em células)	100
AVF (Average Factor)	1/16

Finalmente, se nenhuma destas situações ocorrer, a célula BRM é eliminada e a indicação de geração de uma célula BRM e da passagem da primeira célula BRM é actualizada ($GerarBRM = 0$ e $PassarBRM = 0$).

8.5 Resultados de simulação

Nesta secção é realizada a análise de desempenho do algoritmo de consolidação com base nos resultados obtidos por simulação sobre um conjunto de configurações tipo (configurações com e sem fontes VBR, fontes ABR e VBR a iniciarem a sua actividade decaladas no tempo, etc.). Foram consideradas várias configurações *benchmark* [105], em especial a configuração *Parking Lot* e duas configurações em *Cadeia*, utilizada pelo ATM Forum para avaliar os modelos para conexões ponto-a-multiponto, uma vez que permitem estudar e analisar a distribuição da largura de banda pelos VCs em competição quando, por exemplo, os destinos estão geograficamente dispersos, para além de verificar a forma como o algoritmo trata os problemas do *feedback implosion* e do *consolidation noise*.

Os parâmetros dos BTEs para conexões ABR e dos comutadores de todas as configurações consideradas são apresentados na Tabela 8.1[1].

Relativamente aos comutadores assumiu-se que o tráfego ABR é escalonado se-

[1]Para uma explicação mais detalhada dos parâmetros utilizados pelas fontes e pelos comutadores deve-se consultar a Tabela D.1 do Anexo D.

gundo a disciplina de serviço do tipo FIFO; os tamanhos dos *buffers* foram dimensionados de forma a obter uma percentagem de perdas nula. Todas as ligações oferecem a mesma capacidade 155.52 $Mbit/s$ (149.76 $Mbit/s$ úteis, uma vez que é necessário ter em consideração o *overhead* do SDH), enquanto que a distância entre comutadores é de 10 km; pode também ser usado para simular uma WAN, aumentando apenas as distâncias consideradas. Para todos os casos, é assumido que as fontes ABR são do tipo persistente (*greedy*) e transmitem com o débito permitido pela rede (ACR). Todos os fluxos de tráfego ponto-a-multiponto são no sentido raiz-ramos, com a excepção das células BRM que fazem o percurso inverso.

Os resultados serão apresentados na forma de quatro gráficos por comutador: um gráfico referente ao débito permitido a cada uma das fontes (ponto-a-ponto e ponto-a-multiponto) em $Mbit/s$, um gráfico referente à ocupação das filas em cada porta de saída de cada comutador, um gráfico referente ao factor de carga e um gráfico referente à ocupação das ligações entre os comutadores. Com estes resultados é possível analisar a eficiência, a *fairness*, a escalabilidade, a estabilidade, a ocupação das filas de saída, a utilização dos recursos e a adaptação à variação da largura de banda para fontes com vários modelos de tráfego.

8.5.1 Configuração *parking lot*

A topologia apresentada na Figura 8.1 é conhecida como *"Parking lot"*, devido à semelhança com a situação que acontece quando múltiplas filas de carros se juntam na mesma via em direcção à saída, fenómeno que permite ilustrar os problemas de *fairness*. Nesta configuração existem três VCs ABR ponto-a-ponto (U1, U2 e U3), um VC ABR ponto-a-multiponto (M1) e um VC VBR (V1), com origens e destinos assinalados. O tráfego VBR foi modelizado por uma fonte *on-off*, que alterna períodos activos (30 ms) e de repouso (30 ms), com um débito de 100 $Mbit/s$ durante os períodos activos. Os parâmetros de simulação são mantidos constantes ao longo das várias simulações (valores apresentados na Tabela 8.1).

Com esta configuração os problemas são evidenciados devido à diferença das distâncias, o que vai obrigar a tomar decisões nos pontos de agregação (comutador 2 e 3, quando se observa no sentido das fontes para os destinos), de forma a que seja possível obter compromissos na obtenção de bons resultados, sem necessidade de prejudicar

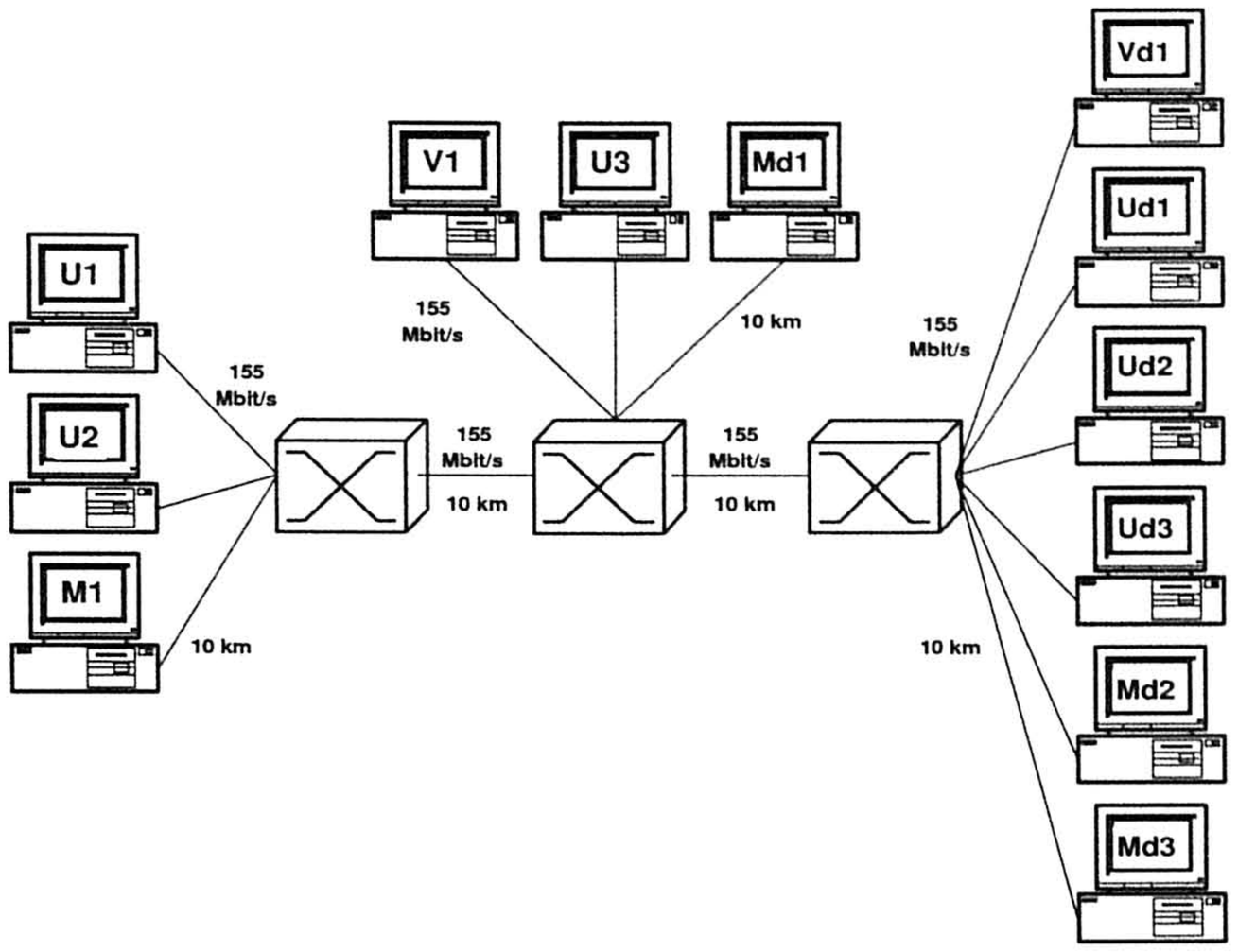

Figura 8.1: Configuração *Parking lot.*

Tabela 8.2: Distribuição da largura de banda pelas fontes ABR nos períodos *On* e *Off* da fonte VBR na configuração *Parking lot.*

Fontes	Largura de banda (*Mbit/s*)	
	VBR *Off*	VBR *On*
U1	37.44	12.44
U2	37.44	12.44
U3	37.44	12.44
M1	37.44	12.44
V1	0.00	100.00

nenhum dos parâmetros de desempenho. A introdução de uma conexão do tipo ponto-a-ponto, mas com uma fonte que gera tráfego prioritário, vai permitir estudar todos os problemas de convergência e de recuperação do sistema para um estado estável, após as perturbações introduzidas no início e no fim dos períodos de actividade dessa fonte.

Os resultados obtidos com a configuração apresentada na Figura 8.1 estão de acordo com os valores da Tabela 8.2, onde é mostrada a distribuição óptima (*max-min fairness*) da largura de banda pelas várias fontes ABR, durante os períodos *On* e *Off* da fonte VBR.

Os resultados estão representados nos gráficos das Figuras 8.2 e 8.3. O gráfico (d) da Figura 8.2 mostra a evolução do factor de carga no primeiro comutador. Dado que a distribuição dos débitos é influenciada pelo congestionamento a jusante, o primeiro

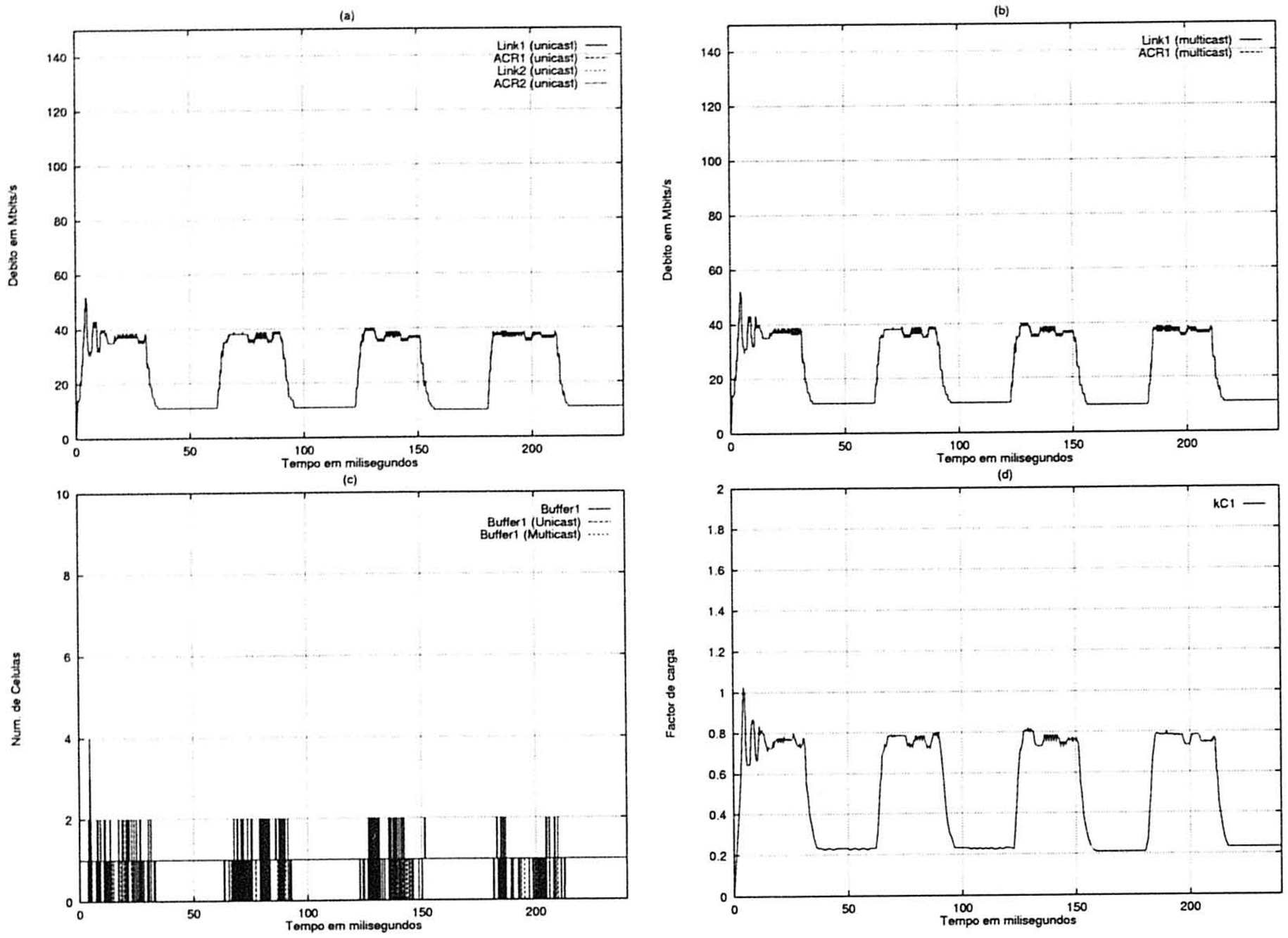

Figura 8.2: Resultados obtidos no primeiro comutador da configuração *Parking Lot*.

comutador não está congestionado (utilização próxima dos 75%), o que é confirmado pela reduzida ocupação da fila de espera, ilustrada no gráfico (c).

Relativamente aos resultados obtidos no segundo comutador (ver Figura 8.3) estes ilustram, como será demonstrado de seguida, o bom desempenho quer do algoritmo de distribuição da largura de banda quer do algoritmo de consolidação. Os resultados apresentados nestes gráficos permitem mostrar o bom comportamento de ambos os algoritmos na presença de transitórios. Na primeira situação, em que se verifica um transitório aos 30 ms, o algoritmo de distribuição da largura de banda converge quase de imediato para o valor que obedece ao critério *max-min fairness*. No instante em que a fonte VBR passa ao estado *Off* (60 ms) é necessária uma redistribuição da largura de banda para que não exista desperdício de recursos, verificando-se igualmente uma rápida convergência para os valores esperados; situação que se vai repetir para cada um dos períodos de transmissão provocados pela fonte VBR.

Para uma melhor compreensão dos resultados obtidos no segundo comutador, o gráfico (d) da Figura 8.3 ilustra o comportamento do factor de carga dentro deste comutador. Este parâmetro dá a indicação dos instantes em que a fonte VBR entra

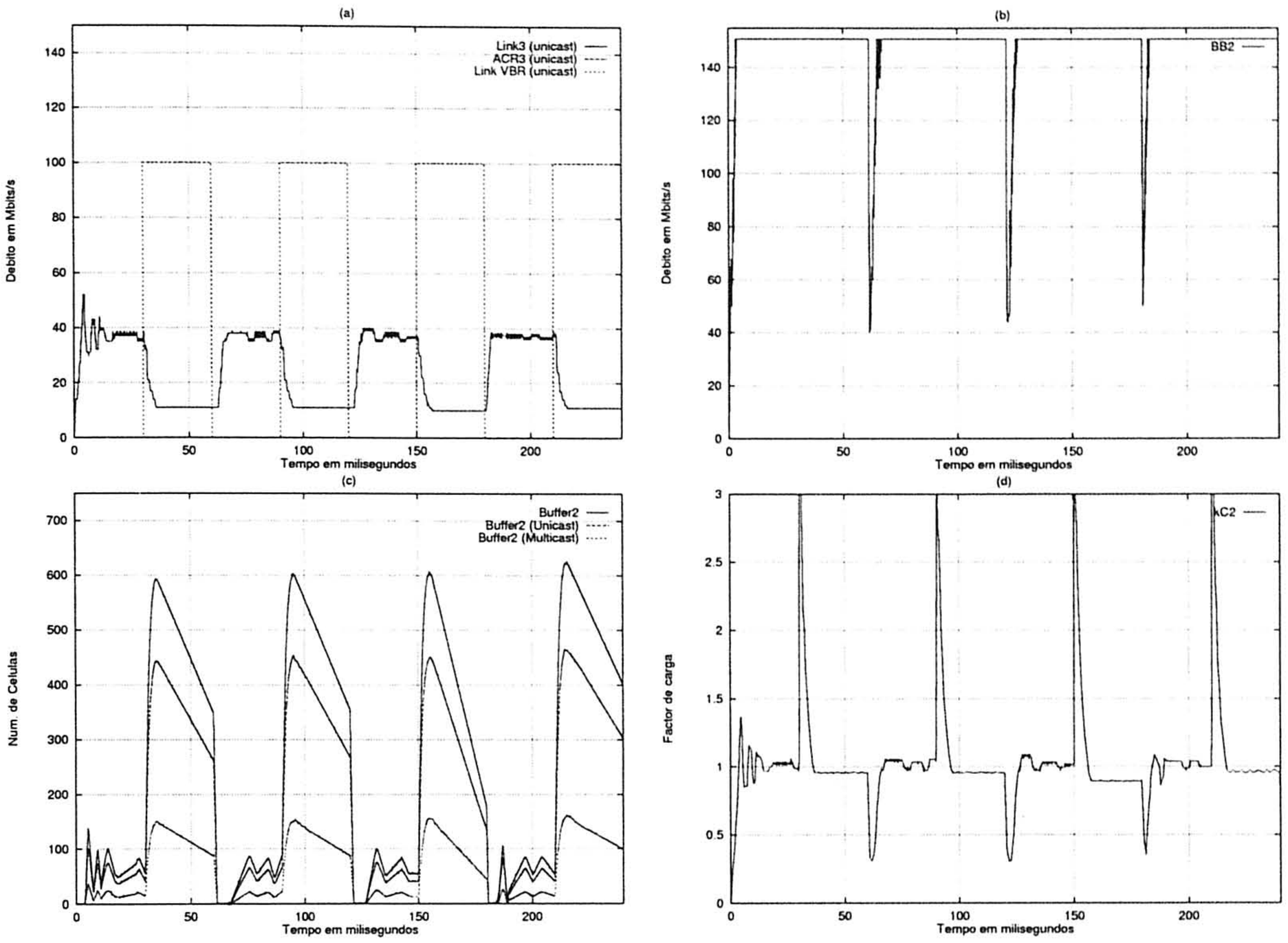

Figura 8.3: Resultados obtidos no segundo comutador da configuração *Parking Lot*.

em actividade ou quando deixa de estar activa e a convergência para valores próximos de 1, a que corresponde a utilização de toda a capacidade ABR disponível. Os valores apresentados no gráfico (c) da mesma figura ilustram a ocupação da fila ABR dentro do segundo comutador. Estes valores servem, a par do factor de carga, para tomar as decisões de quando se deve promover ou reduzir o débito. Nos instantes em que a fonte VBR está no seu período Off, a ocupação mantém-se entre os limiares definidos inicialmente. Este fenómeno observa-se, por exemplo, entre os 70 e os 90 ms. É ainda possível observar que quando a fonte VBR inicia a sua actividade, a ocupação sobe para cerca de 600 células, mas com uma recuperação quase de imediato.

Verifica-se ainda que após ocorrer congestionamento, a recuperação da fonte *multicast* é realizada ao mesmo tempo que a das fontes *unicast*, devido à forma como o algoritmo de consolidação foi concebido: assim que a ocupação da fila de espera ultrapassa o limiar superior, o algoritmo gera uma célula BRM, indicando o estado de congestionamento e o valor do novo débito.

Por último, o gráfico (b) da Figura 8.3 apresenta apenas a ocupação da ligação entre o segundo comutador (congestionado) e o comutador final. A ocupação é próxima

dos 100%, excepto no momento em que a fonte VBR cessa a sua actividade; esta situação deve-se apenas à forma como se realiza o aumento progressivo do débito das fontes, de acordo com o algoritmo normalizado pelo ATM Forum.

8.5.2 Configuração em *Cadeia*

A configuração apresentada na Figura 8.4 designa-se por configuração em *Cadeia* e permite observar com maior facilidade o problema *consolidation noise*. Existiu a preocupação de juntar conexões ponto-a-ponto e ponto-a-multiponto para estudar os problemas da configuração anterior, e igualmente o *fairness* na distribuição da largura banda disponível. É uma configuração constituída por uma conexão ponto-a-ponto (U1) e uma conexão ponto-a-multiponto (M1), onde uma das ligações da conexão do destino mais afastado é a ligação congestionada (partilhada pelas conexões dos dois tipos).

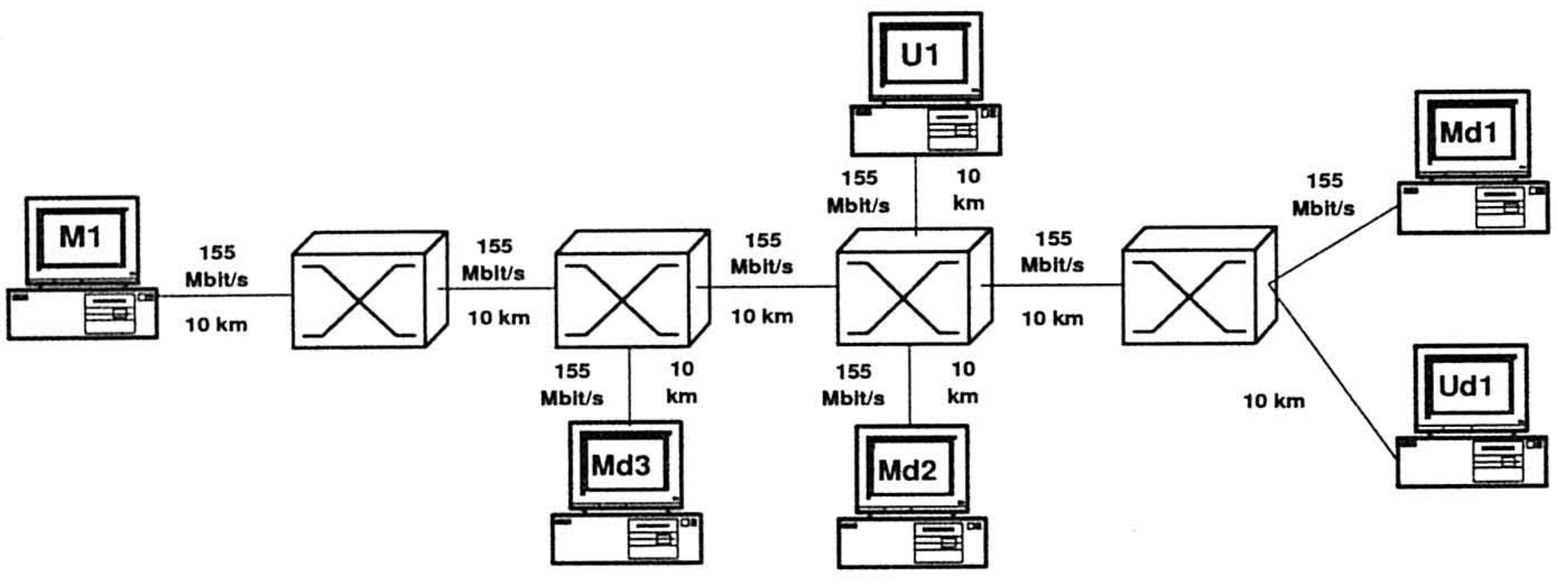

Figura 8.4: Configuração em cadeia.

Os resultados obtidos com a topologia apresentada na Figura 8.4 estão de acordo com os valores da distribuição óptima (*max-min fairness*) da largura de banda pelas duas fontes ABR (75 $Mbit/s$ para cada uma).

O gráfico (d) da Figura 8.5 apresenta a evolução do factor de carga no primeiro e segundo comutador. Dado que a distribuição dos débitos é influenciada pelo congestionamento observado na ligação entre o terceiro e o quarto comutador, o primeiro e

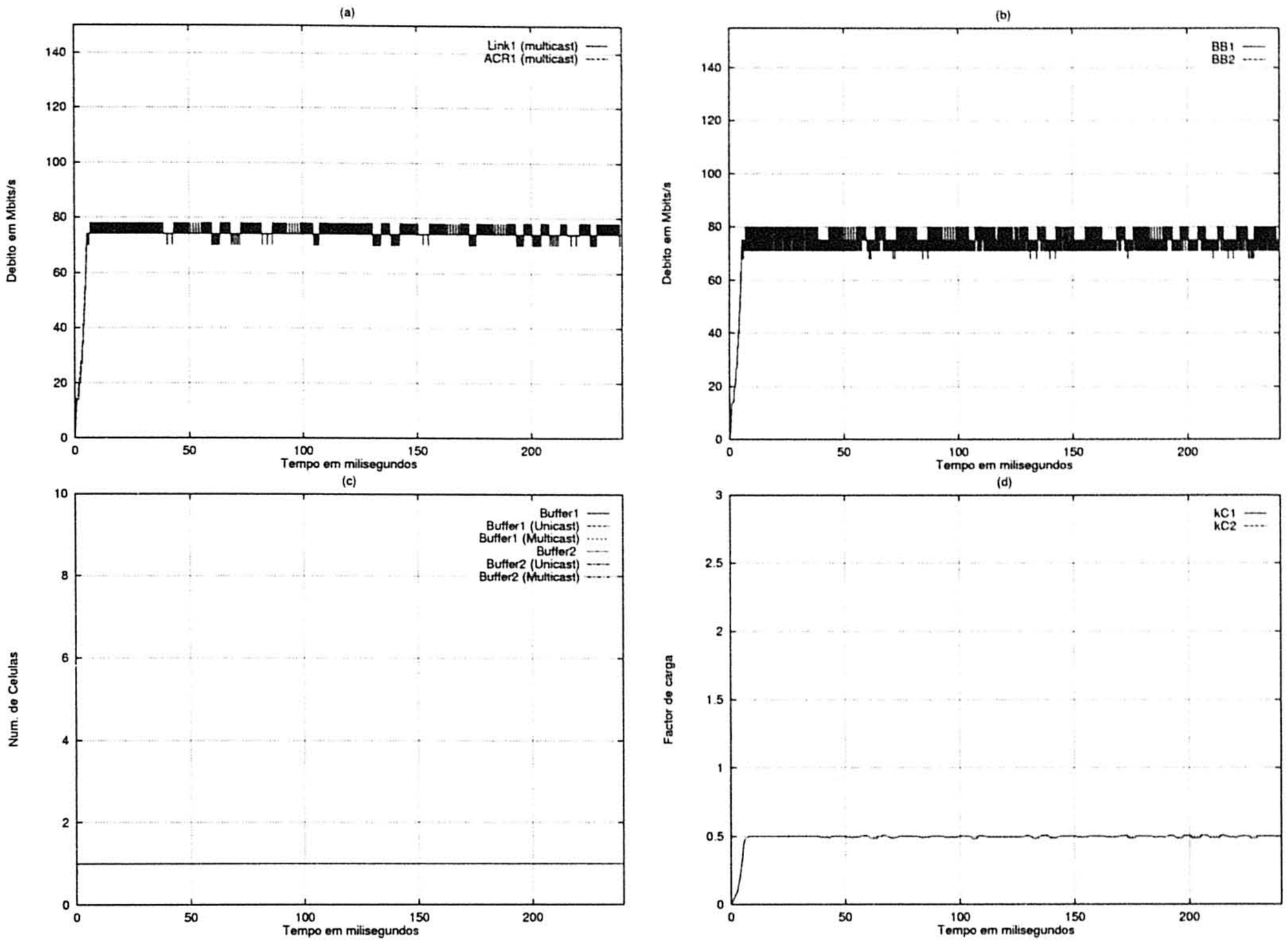

Figura 8.5: Resultados obtidos no primeiro e segundo comutador da configuração em *Cadeia*.

o segundo comutador não estão congestionados (utilização próxima dos 50%), o que é confirmado pela ocupação praticamente nula da fila de espera, ilustrada no gráfico (c).

Relativamente aos resultados obtidos no terceiro comutador, os gráficos da Figura 8.6 permitem observar uma ocupação de 100% (gráfico (b)), a distribuição da largura de banda está de acordo com os valores óptimos (gráfico (a)), cerca de 75 $Mbit/s$, e a ocupação do *buffer* apresenta o padrão de comportamento estabelecido inicialmente, isto é, a sua ocupação varia entre os limiares (Q_H e Q_L) previamente definidos.

8.5.3 Configuração em *Cadeia* com perturbações

Com o mesmo propósito da configuração anterior, a configuração em *Cadeia* com perturbações vai permitir acentuar os problemas *consolidation noise* e *feedback implosion*, porque foi acrescentada mais uma fonte ABR ponto-a-ponto e uma fonte VBR (também ponto-a-ponto), com os mesmos valores utilizados na configuração *Parking Lot*.

Esta configuração (ver Figura 8.7) consiste em quatro comutadores, onde o destino $Md1$ é o que se encontra mais afastado, o destino $Md2$ mais próximo e o destino

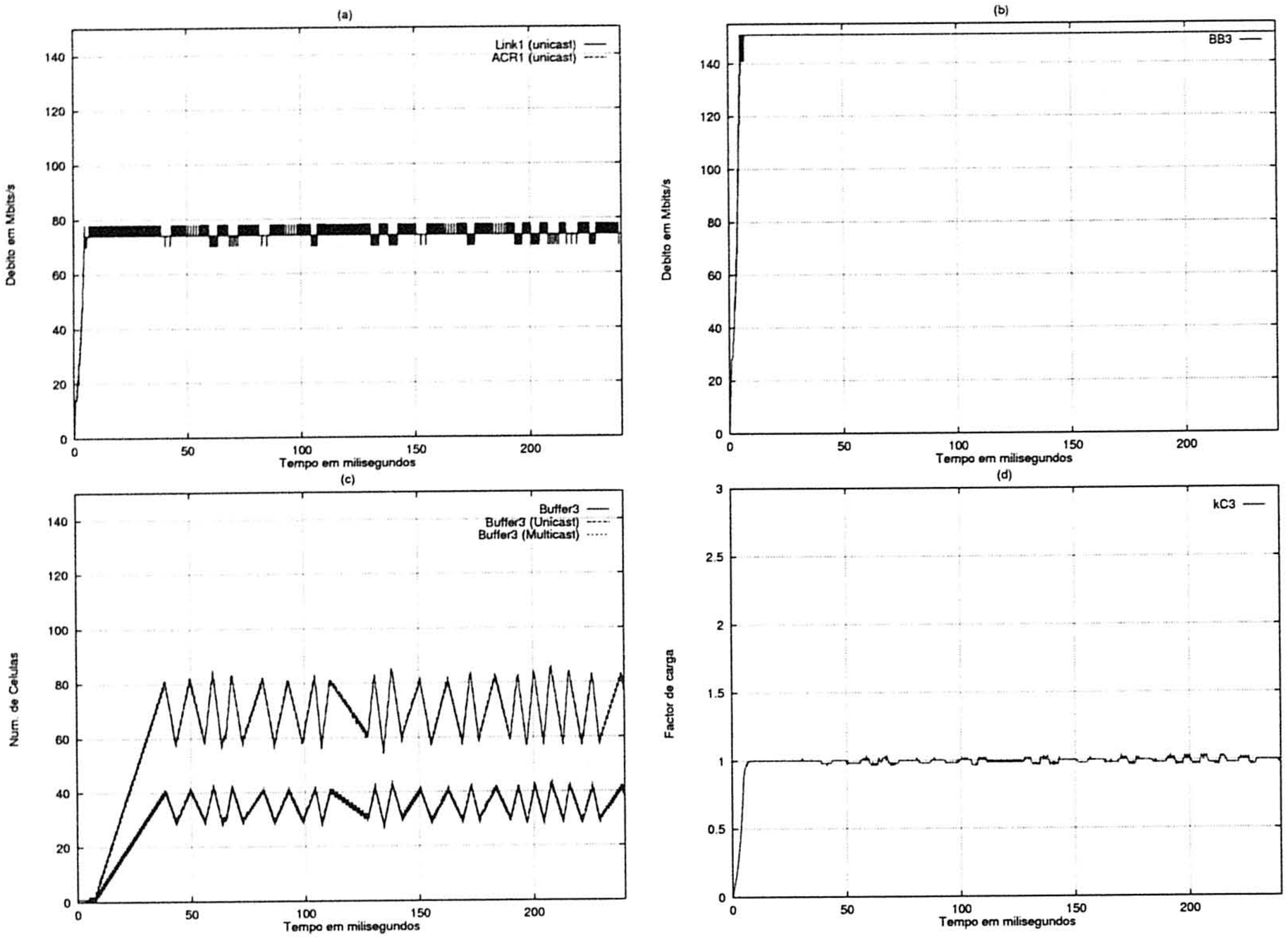

Figura 8.6: Resultados obtidos no terceiro comutador da configuração em *Cadeia*.

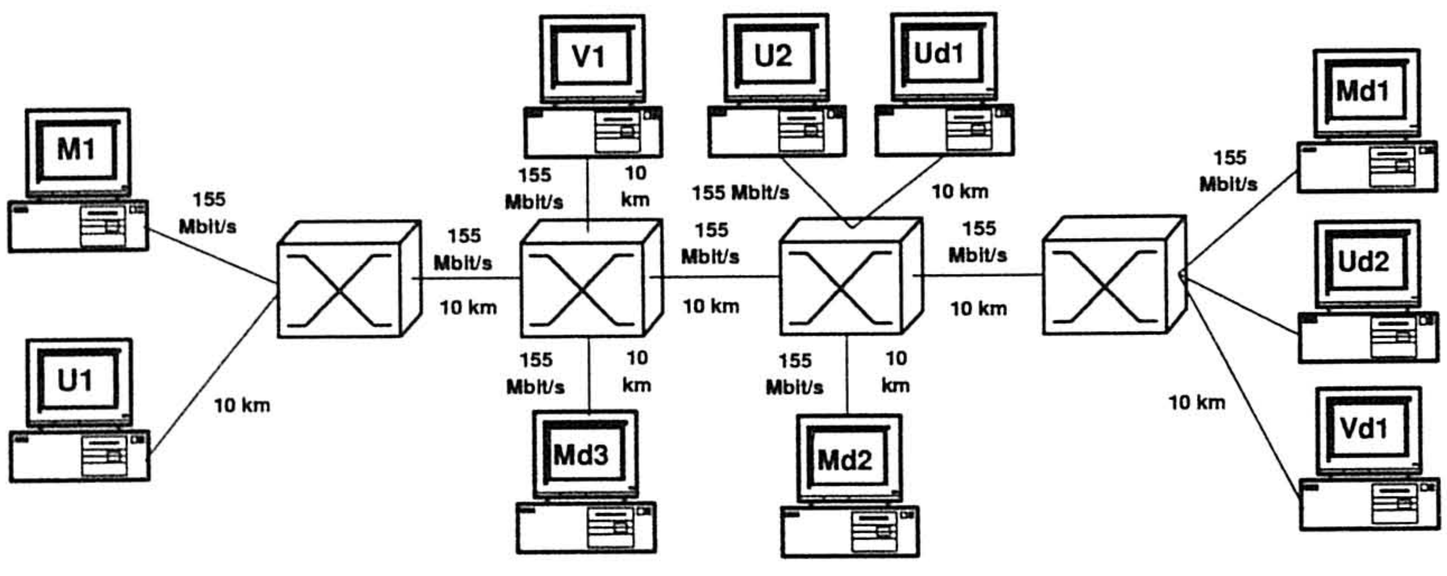

Figura 8.7: Configuração em cadeia com uma fonte VBR.

$Md3$ ainda mais próximo da fonte. É ainda utilizada uma conexão ponto-a-ponto que partilha a ligação com o último destino.

Os resultados obtidos com a configuração apresentada na Figura 8.7 estão de

Tabela 8.3: Distribuição da largura de banda pelas fontes ABR nos períodos *On* e *Off* da fonte VBR na configuração em *Cadeia* com perturbações.

Fontes	Largura de banda (*Mbit/s*)	
	VBR *Off*	VBR *On*
U1	75.00	25.00
U2	75.00	25.00
M1	75.00	25.00
V1	0.00	100.00

acordo com os valores da Tabela 8.3 onde é mostrada a distribuição óptima (*max-min fairness*) da largura de banda pelas várias fontes ABR, durante os períodos *On* e *Off* da fonte VBR.

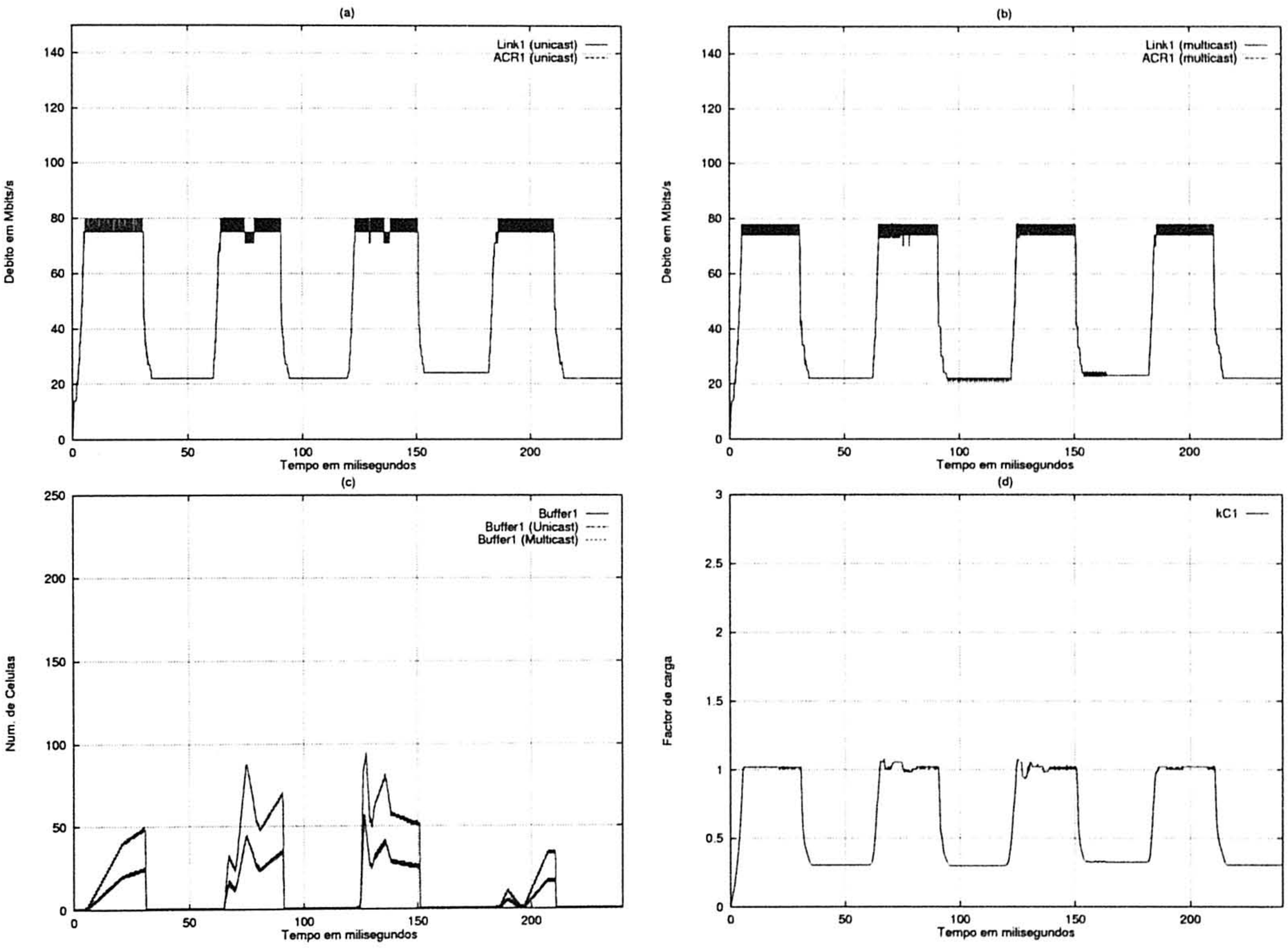

Figura 8.8: Resultados obtidos no primeiro comutador da configuração em *Cadeia* com perturbações.

Os resultados apresentados na Figura 8.8 mostram a evolução do factor de carga (gráfico (d)) no primeiro comutador, a ocupação do *buffer* ABR (gráfico (c)) e a distribuição da largura de banda que é influenciada pelo congestionamento a jusante (a fonte VBR é a única que está ligada ao segundo comutador). Esta influência verifica-se nos dois tipos de conexão, como pode ser observado na distribuição da largura de banda

que está de acordo com os valores óptimos apresentados na Tabela 8.3. É ainda possível perceber que existiu um controlo relativamente à ocupação do *buffer* deste comutador, isto é, a variação da ocupação nos instantes em que a fonte VBR está em repouso apresenta um padrão de comportamento semelhante ao proposto pelo algoritmo de controlo utilizado. Verifica-se ainda que após ocorrer congestionamento, a recuperação da fonte *multicast* é realizada ao mesmo tempo que a das fontes *unicast*, devido à reacção do algoritmo de consolidação.

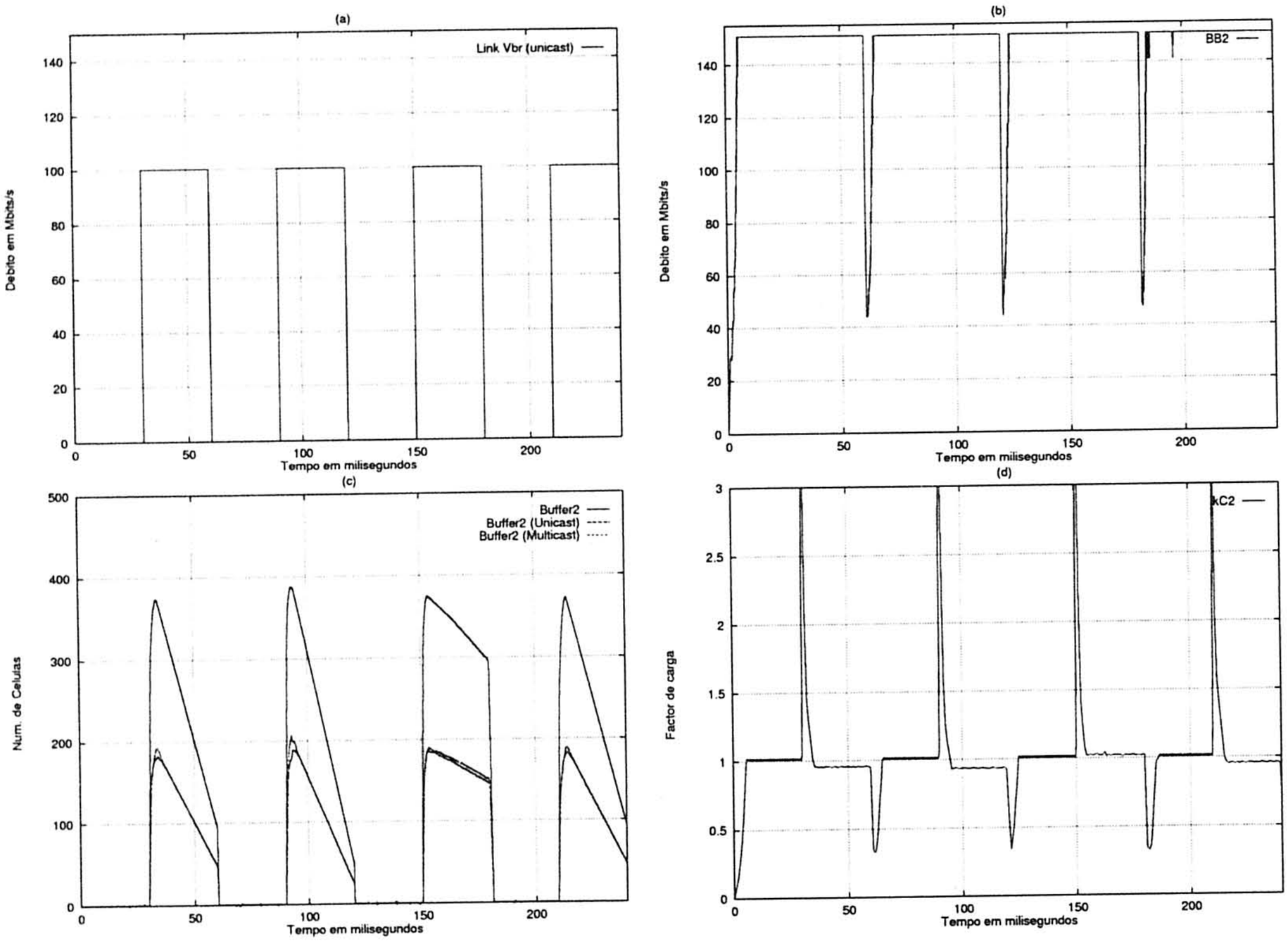

Figura 8.9: Resultados obtidos no segundo comutador da configuração em *Cadeia* com perturbações.

Os resultados obtidos no segundo comutador (ver Figura 8.9) têm uma particularidade que é a ligação da fonte VBR como única fonte que influencia a configuração nos instantes em que está a transmitir. É possível verificar em qualquer um dos gráficos desta figura essa influência, em especial no comportamento dos valores apresentados no gráfico (d) que representa o valor do factor de carga (k) e, como é fácil de observar, evolui em torno de 1 quando está em repouso, tem um aumento significativo no início da actividade e uma redução no instante em que passa ao estado *off*. Os valores apresentados no gráfico (c) da mesma figura ilustram a ocupação da fila ABR e servem, a par do

factor de carga, para tomar as decisões de quando se deve promover ou reduzir o débito. Nos instantes em que a fonte VBR está no seu período *off*, a ocupação mantém-se nula porque não existe outro tipo de fonte ligada ao comutador.

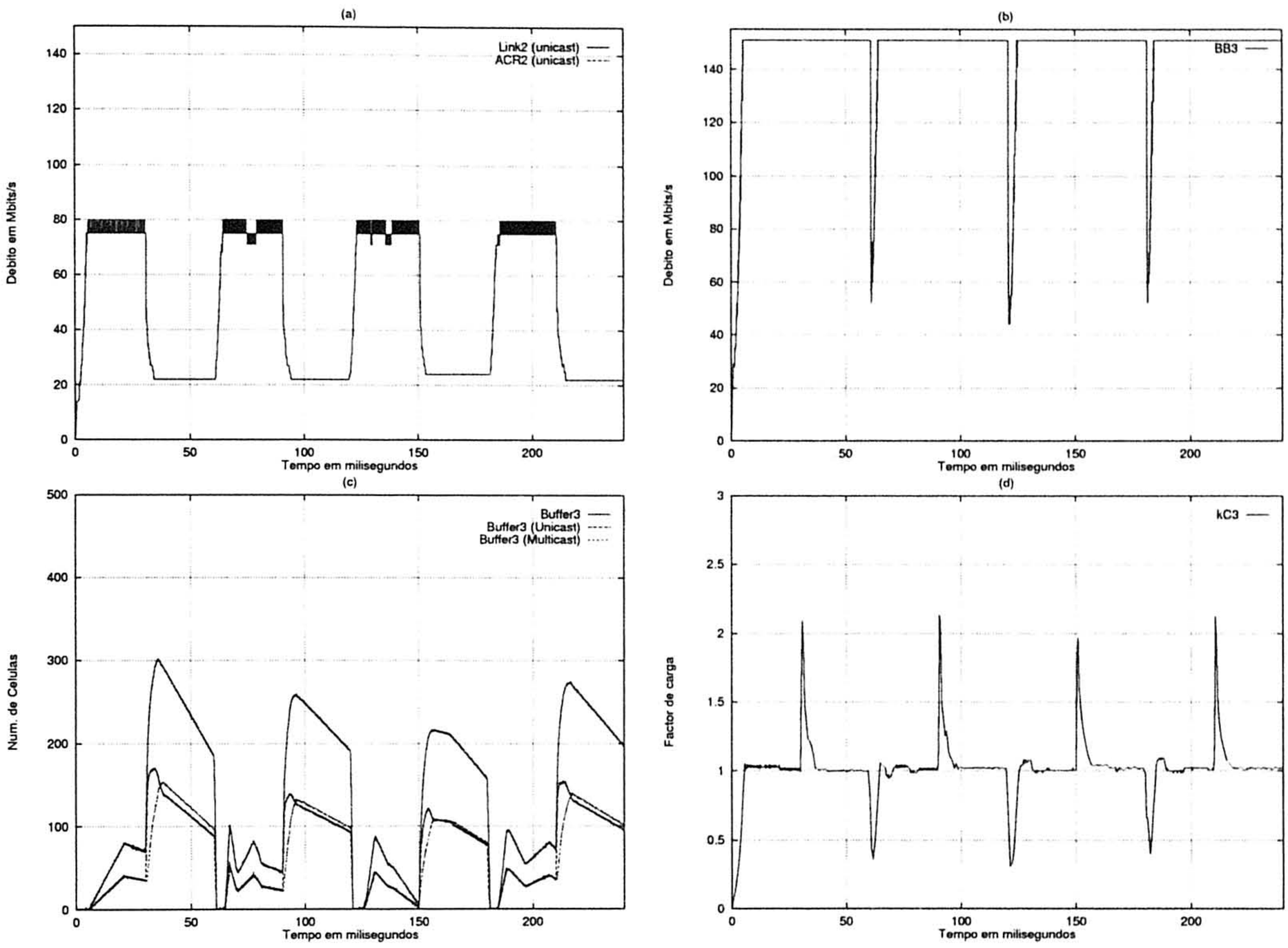

Figura 8.10: Resultados obtidos no terceiro comutador da configuração em *Cadeia* com perturbações.

Por último, os gráficos da Figura 8.10 apresentam a ocupação da ligação entre o terceiro comutador (congestionado) e o comutador final. A ocupação é próxima dos 100%, excepto no momento em que a fonte VBR cessa a sua actividade; esta situação deve-se apenas à forma como se realiza o aumento progressivo do débito das fontes, de acordo com o algoritmo normalizado pelo ATM Forum. Mais uma vez, a largura de banda atribuída à fonte ponto-a-ponto ARB é semelhante aos valores óptimos calculados inicialmente. Da mesma forma, o comportamento do padrão de ocupação no *buffer* (gráfico (c)) encontra-se dentro do definido aquando da apresentação do algoritmo de controlo de fluxo e de congestionamento.

8.6 Conclusões

Nesta secção foi proposto um novo algoritmo de consolidação de células de gestão de recursos para conexões ABR ponto-a-multiponto. Esta solução revelou-se efectiva do ponto de vista da eliminação dos problemas inicialmente apresentados, nomeadamente o *consolidation noise* e o *feedback implosion*, sem impor alterações do formato das células RM, nem dos algoritmos da fonte e do destino, como verificado em algumas das propostas apresentadas na literatura.

Uma das primeiras questões a analisar é a complexidade de implementação, uma vez que um dos objectivos principais deste tipo de algoritmos é apresentar uma complexidade reduzida. Nesta proposta foi necessário manter o número de ramos pertencentes à árvore *multicast* para comparar com o número total de células BRM recebidas de jusante, manter um *bit* por cada porta de saída para verificar se foi recebida uma célula BRM desse ramo, armazenar os valores de MER e as suas comparações com os valores ERQ calculados pelo algoritmo de controlo de fluxo e de congestionamento e os recebidos de jusante ($BRM.ER$). Este algoritmo reduz a complexidade de implementação em relação ao proposto em [36] porque este chama o algoritmo ERICA para todos os ramos, sempre que uma célula BRM é recebida e não apenas quando uma célula BRM está para ser enviada, enquanto que o algoritmo apresentado é menos complexo porque apenas usa os valores obtidos com o algoritmo de controlo de fluxo e de congestionamento [81, 82] quando uma célula BRM é enviada para montante.

A segunda questão que necessita de ser avaliada é o tempo e a forma de resposta ao congestionamento porque a espera de todas as células BRM pode levar a um tempo de resposta lento o que é especialmente grave em casos em que existem poucas ou nenhumas células RM na rede e, por isso, o *feedback* pode ser atrasado em função do maior RTT. O tempo de resposta é mais rápido quando o congestionamento é detectado e comunicado. Por exemplo, na solução proposta em [36] é apenas realizado de uma forma alternativa e depende do algoritmo de controlo de fluxo e de congestionamento (ERICA), enquanto que no algoritmo proposto neste trabalho, esta operação pode ser realizada de duas formas e não depende directamente do algoritmo de distribuição de largura de banda.

A eliminação do problema *consolidation noise* resulta da espera das células BRM de todos os ramos, embora o algoritmo não espere por todas no caso de congestiona-

mento; isto não introduz qualquer problema porque para além das células adicionais segue sempre uma célula BRM, após a recepção de pelo menos uma célula de BRM de cada ramo. A resolução desta questão resulta na introdução de um *overhead* adicional pelas células BRM geradas e enviadas para montante. Como o número de células FRM geradas pela fonte e o número de células BRM recebidas deve ser aproximadamente o mesmo, o algoritmo resolve este problema através do envio de uma célula BRM para montante se o congestionamento teve início ou se se verificou, sendo este um número negligenciável; por exemplo, nos resultados de simulação apresentados na Figura 8.10 foi necessário enviar para montante mais 16 células BRM em 240 ms do que seria de esperar quando a fonte em média gerou 28296 células das quais 884 são FRM, resultando num *overhead* de 1.8%, o que é um valor reduzido, enquanto que a proposta apresentada em [36] mantem um contador que é incrementado sempre que uma célula BRM extra é passada e decrementado no caso das células BRM transportarem informaçao que não indica congestionamento e se o contador é maior do que zero. Em todos os casos, o valor do contador é sempre pequeno porque depende do tempo de convergência do algoritmo ERICA, sendo uma desvantagem porque não permite utilizar outro algoritmo de controlo de fluxo e de congestionamento, mostrando assim que não é independente.

Como foi observado durante a discussão dos resultados, o algoritmo de distribuição da largura de banda em conjunto com o algoritmo de consolidação apresenta bons resultados quanto à reacção aos transitórios, quer na eficiência com que distribuiu a largura de banda quer na rapidez com que se atinge o estado estável. A utilização dos recursos revelou-se eficiente, conseguindo-se simultaneamente o controlo do congestionamento e uma ocupação muito próxima de 100%.

Deve ainda realçar-se a simplicidade do algoritmo, o que torna a sua implementação exequível, constituindo assim uma extensão natural dos algoritmos existentes para ligações ponto-a-ponto.

BIBLIOTECA F.E. PORTO

Capítulo 9

Conclusões e perspectivas futuras

As conclusões de um trabalho com a dimensão e a complexidade próprias de uma tese de doutoramento apresentam-se como um desafio porque, por um lado, é necessário ser-se o mais objectivo possível na apresentação das conclusões sem nunca perder o rigor científico e, por outro lado, é necessário mostrar linhas de investigação para trabalhos posteriores, sem esquecer que as soluções apresentadas devem contribuir de alguma forma para o avanço científico.

No primeiro caso, as conclusões devem envolver um misto de resumo de cada um dos componentes do trabalho desenvolvido de forma a mostrar se os resultados obtidos são de facto condizentes com os objectivos estabelecidos inicialmente, se foram atingidos ou a que "distância" é que ficaram e o porquê dessa não total "satisfação". Neste caso, os objectivos propostos foram praticamente todos atingidos, como pode ser confirmado pelos resultados apresentados e discutidos ao longo dos capítulos. Contudo, quando se utiliza uma expressão do tipo "praticamente" pode-se pensar que não foram todos atingidos na sua totalidade; isto é de certa forma verdade porque a ideia inicial é sempre bastante ambiciosa (às vezes em demasia) e os resultados finais quase sempre ficam a alguma distância. Como o objectivo é ir sempre o mais longe possível, a distância que falta percorrer pode ser apelidada de trabalho futuro ou de perspectivas futuras.

9.1 Conclusões

A ideia de integração total entre redes IP e ATM (de acordo com os respectivos modelos arquitectónicos "nativos"), como objectivo inicial deste trabalho, deixou de o ser à

medida que as tecnologias e trabalhos de investigação foram evoluindo. Deixando de ser prioritário este objectivo inicial, passou-se a pensar na integração entre estas tecnologias de forma diferente, isto é, realizar uma integração através da subdivisão das redes em redes periféricas e redes de núcleo, com mecanismos de interface entre ambas (ED). Este tipo de integração pressupõe uma nova filosofia baseada num único tipo de equipamentos capazes de integrar *routers* (nível 3), com comutação de etiquetas (nível 2). Assim, para as redes periféricas a escolha incidiu sobre as redes com tecnologia IP e para a rede de núcleo escolheu-se a tecnologia ATM[1] com VCs pertencentes à categoria de serviço ABR. A escolha deste tipo de categoria deveu-se às suas características, principalmente o facto da rede fornecer informações sobre o seu estado, como por exemplo, a largura de banda disponível, a indicação de congestionamento, etc.

Após ter-se optado por este tipo de organização e das respectivas redes era necessário estabelecer que tipo de intervenção deveria ser executada e em que pontos é que se justificaria. A primeira situação a estudar seria escolher o protocolo a utilizar na rede de acesso; no entanto, não foi proposta nenhuma solução porque já existem alternativas interessantes. Este protocolo (da rede de acesso) teria de ter a capacidade de fazer as fontes IP adaptarem-se aos débitos fornecidos pela rede de núcleo, de forma a reduzir ao máximo a perda de pacotes e conseguir atingir os requisitos da QoS necessários às aplicações sem uma dependência de protocolos das camadas superiores, isto é, ter o mesmo comportamento para aplicações que utilizam o TCP ou o UDP.

Admitindo que o problema do protocolo da rede de acesso estava resolvido colocava-se agora a situação mais complexa deste trabalho, que era definir uma arquitectura para os EDs e um método para escalonar fluxos IP agregados em VCs ATM, mas de forma a conseguir ser o mais equitativo possível na distribuição da largura de banda, dos atrasos e do controlo da ocupação dos *buffers*. Para resolver este problema a escolha incidiu sobre a utilização de TBs com funções adicionais aquelas para as quais foram desenvolvidos. Foi nesta fase que se tornou necessário conceber um método que conseguisse realizar uma distribuição o mais equitativa possível da largura de banda e dos atrasos e, para isso, foram estudados dois métodos existentes para verificar se conseguiam atingir estes objectivos. Os métodos estudados foram o FIFO e o WFQ que apresentaram os resultados esperados, isto é, o FIFO serve apenas para tráfego

[1]Neste contexto o ATM é utilizado apenas como uma tecnologia de comutação de etiquetas

sem qualquer requisito e o WFQ não consegue garantir os objectivos pré-estabelecidos. Percebendo que os algoritmos existentes não resolviam esta questão foi necessário desenvolver um método de raiz.

A definição do novo método de escalonamento teve por base a utilização de um TB com um *buffer* adicionado a jusante de cada um e desta forma as células (após os pacotes serem segmentados de acordo com o protocolo AAL5) são escalonadas com base no estado dos TBs, dos *buffers* associados e dos créditos atribuídos a cada fonte. Foram estabelecidos novos parâmetros que se podem dividir em dois tipos, os parâmetros relacionados com os TBs (X_i, Z_i, Q_i) e os parâmetros auxiliares utilizados para a distribuição equitativa do serviço ($credits_i$, $credits_i^{acu}$). É possível verificar que os resultados obtidos vão ao encontro dos objectivos estabelecidos inicialmente. Como pode ser constatado através da observação da evolução das soluções propostas, os resultados foram sempre melhorando à medida que uma nova solução era apresentada; no entanto, é necessário realçar que, desde a primeira solução, a equidade na distribuição da largura de banda foi sempre atingida e a ocupação máxima dos *buffers* nunca foi ultrapassada. Nos resultados obtidos com a solução final foram atingidos todos os objectivos propostos.

Estando o problema do escalonamento resolvido era necessário idealizar um esquema para distribuir a largura de banda recebida da rede pelos VCs. Nesta situação era possível utilizar soluções já existentes mas, como os documentos do ATM Forum indicam que estas soluções devem ser específicas das implementações (sugerindo abordagens, mas não impondo soluções), e após se ter verificado que existem vários critérios de *fairness* aplicáveis a tráfego ABR com garantias dum débito mínimo (MCR), resolveu-se propor uma nova solução que fosse simples e ultrapassasse limitações das soluções conhecidas.

Assim que a categoria de serviço estava escolhida e o mecanismo de controlo de fluxo e de congestionamento para VCs ponto-a-ponto estava resolvido era necessário avançar para uma solução que melhorasse a gestão dos recursos dentro da rede de núcleo e, por isso realizou-se uma evolução dos VCs ponto-a-ponto para VCs ponto-a-multiponto. Uma das primeiras vantagens desta evolução é a escalabilidade, uma vez que é possível ter apenas um VC deste tipo para uma única fonte e vários destinos. No entanto é possível explorar os outros tipos de conexões multiponto, nomeadamente o multiponto-a-ponto e o multiponto-a-multiponto, mas não foi essa a opção, ficando esta possibilidade para trabalhos futuros. Os VCs ponto-a-multiponto apresentam al-

guns problemas específicos, nomeadamente o *feedback implosion* e o *consolidation noise*. Ambos os problemas foram resolvidos através da proposta realizada. Da mesma forma que nos casos anteriores, todos os objectivos iniciais foram alcançados, como pode ser comprovado pelos resultados apresentados.

Como conclusão final é possível afirmar que este trabalho apresenta contribuições em três áreas diferentes:

1. Apresentação de uma arquitectura dos EDs com a utilização de TBs e de um método de escalonamento que, para além de preocupações na garantida da distribuição da largura de banda pelos fluxos em competição de acordo com os seus requisitos, introduz dois objectivos adicionais que aumentam a complexidade do problema: a distribuição equitativa dos atrasos e do controlo da ocupação dos *buffers*. Os três objectivos foram plenamente atingidos.

2. A segunda contribuição incidiu sobre um mecanismo de distribuição da largura de banda atribuída à categoria ABR, através da definição de um novo critério de *fairness*, incorporado no algoritmo de atribuição de largura de banda implementado nos comutadores.

3. A terceira contribuição baseou-se no desenvolvimento de um algoritmo de consolidação de células de gestão de recursos e distribuição da largura de banda em conexões ponto-a-multiponto e, também nesta situação os objectivos propostos foram todos atingidos.

9.2 Perspectivas futuras

A ideia de uma secção intitulada "Perspectivas futuras" faz todo o sentido porque num trabalho com esta duração temporal, acompanhado quase sempre de ambiciosos objectivos, chega um momento em que se percebe que muita coisa ficou por fazer e que merece uma atenção especial, podendo dar origem a novos trabalhos de investigação.

Antes de apresentar algumas ideias para trabalhos futuros começa-se por indicar a forma como os assuntos serão organizados. Em primeiro lugar serão apresentadas as questões que não foram tratadas durante este trabalho, mas que merecem uma atenção especial, pois a sua resolução poderá contribuir para o estabelecimento de uma solução

mais "completa". Em segundo lugar serão abordados onde foram apresentadas propostas concretas, bem como resultados, mas onde existem aspectos que poderão ser melhorados.

Uma questão que não foi objecto de nenhuma proposta concreta, mas que faz parte de todo um conjunto (que necessita de ser tratado) é a forma como comunicar às fontes IP as alterações relativamente à largura de banda disponível, de forma a que estas se possam adaptar. Nesta área é possível citar um trabalho de doutoramento [55] que se dedicou exclusivamente à sua resolução. Esta situação, segundo o referido autor, vai obrigar a realizar modificações na camada IP das fontes, sendo portanto alterações que requerem cuidado, devido à implantação da tecnologia IP no mundo actual, isto é, qualquer alteração terá de ser pesada face às consequência do seu impacto, para além da sua viabilidade. Contudo, esta questão continua em aberto, uma vez que as fontes deste tipo não foram projectadas para ter este comportamento, quando o tipo de necessidades são bem diferentes daquelas que existiam aquando da introdução do IP.

Com base na ideia explorada em [55] e na filosofia de funcionamento da categoria de serviço ABR é possível apresentar uma sugestão que pode ser explorada no sentido de desenvolver um protocolo que exija o menor número de alterações na camada IP, proporcione um controlo efectivo do débito das fontes, permita uma partilha equitativa da largura de banda pelas fontes concorrentes na mesma conexão e, essencialmente, permita a escalabilidade por exigir um número reduzido de pacotes de controlo na rede.

Intuitivamente, após a definição destes objectivos, alguns ou quase todos estão relacionados com os objectivos propostos pelos esquemas de controlo de débito em conexões ABR, pelo que é possível pensar numa solução similar para a relação entre as fontes IP e os EDs. A ideia desenvolve-se em torno do comportamento típico de uma conexão ABR, isto é, a fonte solicita a transmissão ao débito máximo (PCR) e recebe da rede o débito máximo possível (ER), que deve ser superior ou igual ao valor mínimo previamente definido (MCR) [42]. Com base neste comportamento, um esquema possível seria o de transmitir pacotes IP intercalados com pacotes IP de gestão de recursos (*Resource Management IP*, RMIP, por exemplo), que transportassem o débito máximo pretendido, o endereço IP da fonte, o endereço IP do ED e o débito corrente. Contudo, como ao nível IP existe multiplexagem dos fluxos TCP/UDP é necessário discriminar um identificador adicional; por exemplo, o endereço IP mais o porte, ou *flow label* do IPv6, etc. O ED, com base no valor do débito calculado nas portas de entrada,

em conjunto com o débito agregado à entrada das filas de espera do VC correspondente e com o débito de saída para a conexão ABR, enviará o débito obtido através de um pacote RMIP do mesmo tipo; no entanto, este pacote só será retornado se existir alguma diferença entre o débito corrente e o débito disponível na conexão ABR atribuída a essa fonte, de contrário o pacote é eliminado.

Esta ideia parece ser simples, mas necessita de alguma reflexão quanto ao tipo de pacote IP (por exemplo, ICMP com outro tipo de função) a gerar nas fontes, qual a periodicidade desses pacotes (por número, ou por tempo, ou pela solicitação da própria aplicação, etc.), para além de outras questões. Além dessa reflexão, é ainda necessário perceber que tipo de modificações é necessário realizar nas fontes, bem como o impacto dessas modificações, por um lado, quanto à complexidade introduzida e, por outro lado, quanto a forma como elas se iriam propagar pela grande quantidade de máquinas IP existentes. Esta última preocupação deve ser ponderada porque basta ver o que está acontecer com o IPv6 que ainda não está generalizado quando a versão corrente já tem o seu espaço de endereçamento praticamente esgotado [119]. A largura de banda ocupada por este tipo de pacotes pode ser considerada negligenciável devido ao seu tamanho reduzido, não esquecendo, no entanto, que o tipo de modelo adoptado terá sempre influência na percentagem de ocupação (à semelhança da taxa de ocupação das células RM).

Esta sugestão fica para um estudo posterior, uma vez que é assumido que o débito é comunicado às fontes para que estas se possam adaptar. Contudo, este problema não é tratado no presente trabalho, porque admitiu-se que é possível utilizar qualquer uma das técnicas e modelos na rede de acesso deixando a preocupação fundamental para o comportamento dos EDs.

Dentro dos assuntos estudados existem situações que poderão ter um tratamento diferente, nomeadamente as opções tomadas quanto ao escalonamento de fluxos de tráfego IP através da utilização de TBs, e da técnica do cálculo de créditos a atribuir a cada um dos fluxos. A questão que se pode levantar é a utilização de algoritmos de escalonamento para o método proposto, mas baseados em diferentes técnicas; esta é uma questão que pode e deve ser explorada futuramente.

Relativamente ao algoritmo de controlo de fluxo e de congestionamento da categoria de serviço ABR para conexões ponto-a-ponto (ou ponto-a-multiponto), uma das

questões que poderão ser abordadas em trabalhos futuros tem a ver com a sua justificação e a análise teórica do desempenho do algoritmo proposto para consubstanciar os resultados obtidos por simulação, como apresentado em [23] ou [7].

Quanto à escolha da tecnologia da rede de núcleo é possível especular em torno de outras tecnologias, por exemplo o MPLS ou até a utilização da combinação do ATM e do MPLS para suportar os modelos IP com QoS (*IntServ*, *DiffServ*) e o *Best-Effort*. A ideia de utilizar as duas tecnologias na rede de núcleo pode ser interessante, quando observada do ponto de vista das capacidades fornecidas por cada uma delas, isto é, o MPLS pode ser utilizado para transportar, por exemplo tráfego *DiffServ* e *Best-Effort* e o ATM para transportar tráfego *IntServ* devido à sua elevada capacidade de gestão de largura de banda, reduzidos atrasos extremo-a-extremo, etc. Contudo, é possível questionar a utilização ou não do ATM para suportar *DiffServ*. A resposta pode ser positiva ou negativa, dependendo dos argumentos apresentados; por exemplo a negação da utilização desta combinação encontra justificação na dificuldade de mapeamento dos níveis de eliminação dos pacotes IP porque quando a tecnologia que suporta *DiffServ* é ATM é possível apenas suportar dois níveis de eliminação o que se torna uma limitação.

Se a resposta é positiva, isto é, utilizar a combinação de ambas (*DiffServ*/ATM), então é possível mapear os seviços definidos no *DiffServ* numa rede ATM; no entanto é necessário fazer a distinção entre serviços relativos e absolutos em relação à QoS, uma vez que a QoS numa conexão ATM é definida no início e mantida até ao fim e num fluxo IP pode sofrer alterações. Assim, uma possível solução é mapear uma classe de serviço do *DiffServ* com QoS (EF e AF) numa categoria de serviço ATM. Se o objectivo for suportar serviços absolutos, a escolha dos serviços a mapear deve ser baseado nas medidas de QoS, por exemplo, um serviço IP com requisitos quanto aos atrasos (baseado nos *DiffServ*) pode ser mapeado numa categoria ATM, de acordo com os parâmetros de tráfego e características de QoS. Admitindo que existe uma aplicação de vídeo e que utiliza a classe AF do *DiffServ* é possível, por exemplo, mapear esta classe nas categorias de serviço rt-VBR ou ABR, desde que exista uma correspondência entre os parâmetros específicos da aplicação e os parâmetros PCR, MBS e SCR, no caso do rt-VBR e o MCR no ABR. Quanto aos serviços relativos, estes podem ser mapeados na categoria de serviço UBR como apresentado em [43].

A escolha das conexões ponto-a-multiponto deveu-se às necessidades impostas

por aplicações que têm este requisito e, por isso, apresentou-se um algoritmo de consolidação de células de gestão de recursos que circulam no sentido destinos-fonte, uma vez que não existe nenhuma norma proposta pelo ATM Forum. Relativamente às conexões multiponto-a-ponto os problemas são distintos dos do tipo de conexões precedente, nomeadamente o *VC-merge* (com implicações directa no *cell interleaving*), uma vez que se está a utilizar células do tipo AAL5 que não possuem mecanismos de controlo, nem número de sequência, o que obriga a usar técnicas auxiliares para conseguir contornar este problema, para além de não existir nenhuma norma proposta pelo ATM Forum, logo justifica-se a apresentação de novas soluções. Se não existem normas para os dois tipos de conexões anteriores, muito menos existe para as conexões multiponto-a-multiponto. A par dos problemas de controlo de fluxo e congestionamento deste tipo de conexões é necessário observar que se colocam ainda os problemas de sinalização, de encaminhamento, etc., ficando assim uma grande área de trabalho em aberto à espera de novas contribuições.

Bibliografia

[1] H. F. Badran and H. T. Mouftah. ATM Switch Architecture with Input-Output Buffering: Effect of Input Traffic Correlation, Contention Resolution Policies, Buffer Allocation Strategies and Delay Backpressure Signal. *Computer Networks ISDN Systems*, 26(9):1187–1213, May 1994.

[2] H. Balakrishnan, H. S. Rahul, and S. Scshan. An Integrated Congestion Management Architecture for Internet Hosts. *in Proceedings of ACM SIGCOMM*, September 1999.

[3] A. W. Barnhart. Example Switch Algorithm for TM Spec. In *ATM Forum 95-0195*, February 1995.

[4] J. C. R. Bennett and H. Zhang. WF2Q: Worst-case Fair Weighted Fair Queueing. *in Proceedings of IEEE Infocom '96*, March 1996.

[5] J. C. R. Bennett and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. *IEEE/ACM Transactions on Networking*, October 1997. Also in Proceedings of IEEE Infocom 96, 1996.

[6] S. Berson. Ip Integrated Services Support in ATM. *Internet Draft*, draft-ietf-issl-atm-support-00, June 1996.

[7] S. Bhatnagar, M. C. Fu, and S. I. Marcus. Optimal Structured Feedback Policies for ABR Flow Control Using Two-Timescale SPSA. *IEEE/ACM Transactions on Networking*, 9(4):479–491, August 2001.

[8] A. Birman, H. R. Gail, S. L. Hantler, Z. Rosberg, and M. Sidi. An Optimal Service Policy for Buffer Systems. *Technical Report TR*, March 1993.

[9] F. Bonomi and K. W. Fendick. The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service. *IEEE Network Magazine*, pages 25–39, March/April 1995.

[10] J. Y Le Boudec. Application of Network Calculus to Guaranteed Service Network. *IEEE Transactions On Information Theory*, 44(3):1087–1096, May 1998.

[11] J.-Y. Le Boudec and P. Thiran. A Short Tutorial on Network Calculus I: Fundamental Bounds in Communication Networks. *Proceedings ISCAS2000*, Geneve, Switzerland, May 2000.

[12] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministc Queuing Systems for the Internet.* Springer-Verlag Lecture Notes on Computer Science 2050, first edition, 2001.

[13] J.-Y. Le Boudec, P. Thiran, and S. Giordano. A Short Tutorial on Network Calculus II: Min-Plus Theory to Communication Networks. *Proceedings ISCAS2000*, Geneve, Switzerland, May 2000.

[14] C. Brown. Requirements for the MPOA Protocol. *ATM Forum 95-0004R2*, April 1995.

[15] C. Brown. Baseline Text for MPOA. *ATM Forum 95-0824R6*, February 1996.

[16] M. M. Buddhikot and C. Papadopoulos. Washington university workshop on integration on IP and ATM: Overview of session 3: Multicast support in ATM+IP networks. April 1996.

[17] A. Charny, D. D. Clark, and R. Jain. Congestion Control with Explicit Rate Indication. In *Proceedings of the IEEE International Communications Conference (ICC)*, June 1995.

[18] Anna Charny. An Algorithm for Rate Allocation in a Cell-Switching Network with Feedback. Master's thesis, Massachusetts Institute of Technology, May 1994.

[19] H-S. A. Chen and K. Nahrstedt. Feedback Consolidation and Timeout Algorithms for Point-to-Multipoint ABR Service. Technical report, Technical Re-

port, UIUCDCS-R-99-2083, UILU-ENG-99-1704, Department of Computer Science, University of Illinois, at Urbana - Champaign, February 1999.

[20] H-S. A. Chen and K. Nahrstedt. Feedback Consolidation and Timeout Algorithms for Point-to-Multipoint ABR Service. In *Proceedings of the International Conference on Communications (ICC '99), http://monet.cs.uiuc.edu/ chen5/R992083.ps.gz*, June 1999.

[21] J. S. C. Chen and T. E. Stern. Throughput Analysis, Optimal Buffer Allocation, and Traffic Imbalance Study of a Generic Nonblocking Packet Switch. *IEEE JSAC*, 9(3):439–449, April 1991.

[22] F. M. Chiussi, Y. Xia, and V. P. Kumar. Dynamic Max Rate Control Algorithm for Available Bit Rate Service in ATM Networks. In *Proceedings of GLOBECOM '96, volume 3, London, UK, pp. 2108-2117*, November 1996.

[23] S. Chong, S. Lee, and S. Kang. A Simple, Scalable, and Stable Explicit Rate Allocation for MAX-MIN Flow Control with Minimum Rate Guarantee. *IEEE/ACM Transactions on Networking*, 9(3):322–335, June 2001.

[24] The ATM Forum Technical Committee. ATM User-Network Interface (UNI) Signalling Specification Version 4.0. *ATM Forum*, af-sig-0061.000, July 1996.

[25] R. L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *in IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[26] R. L. Cruz. A Calculus for Network Delay, Part II: Network Analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[27] R. L. Cruz. Quality of Service Guarantees in Virtual Circuit Switched Networks. *in IEEE Journal on Selected Areas in Communications*, 13(6):1048–1056, August 1995.

[28] B. Davis, P. Doolan, and J. Lawrence et all. Use of Tag Switching With ATM. *Internet Draft*, draft-davie-tag-switching-atm-01, January 1997.

[29] S. Deering. Host Extension for IP Multicasting. *RFC 1112*, August 1989.

[30] A. Demers, S. Keshav, and S. Shenker. Analisys and Simulation of a Fair Queueing Algorithm. *In Jornal of Internetworking Research and Experience*, pages 3–26, October 1990.

[31] N. Demizu, K. Nagami, P. Doolan, and H. Esaki. VCID: Virtual Connection Identifier. *Internet-Draft, draft-demizu-mpls-vcid-01*, October 1997.

[32] S. Blake et al. An Architecture for Differentiated Services. In *IETF RFC 2475*, December 1998.

[33] B. Vandalore S. Fahmy, R. Jain, R. Goyal, and M. Goyal. A Definition of General Weighted Fairness and its Support in Explicit Rate Switch Algorithms. *in Proceedings of ICNP'98*, pages 22–30, October 1998.

[34] S. Fahmy and R. Jain. ABR Flow Control for Multipont Connections. *IEEE Networks*, 12(5):6–7, September/October 1998.

[35] S. Fahmy, R. Jain, and et. al. Performance analysis of ABR point-to-multipoint connections for bursty and non-bursty traffic with and without vbr background. *ATM Forum/97-0422*, April 1997.

[36] S. Fahmy, R. Jain, and et. al. Feedback Consolidation Algorithms for ABR Point-to-Multipont Connections in ATM Networks. *Proceedings of INFOCOM'98, San Francisco, CA*, 3:1004–13, March 1998.

[37] S. Fahmy, R. Jain, S. Rabie, R. Goyal, and B. Vandalore. Quality of Service for Traffic over ATM Service Categories. *To appear in Computer Communications*, 1999.

[38] D. Ferrari and D. Varma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE JSAC*, 8(3):368–379, April 1990.

[39] D. Ferrari and D. C. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *Jornal of High Speed Networks*, 1(1), 1992.

[40] ATM Forum. LAN Emulation over ATM, Version 1.0. In *Foster City, CA*, January 1995.

[41] ATM Forum. Traffic Management Specification Version 4.0. In *ATM Forum 95-0013R10*, February 1996.

[42] ATM Forum. The ATM Forum traffic management Specification version 4.1. *ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0121.000.pdf*, March 1999.

[43] ATM Forum. Enhancements to support IP Differentiated Services on IEEE 802.1D over ATM. *BTD-TM-DIF-01.04 draft*, May 2000.

[44] The ATM Forum. The ATM Forum traffic management Specification version 4.0. *ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps*, April 1996.

[45] E. Gauthier, J.-. Le Boudec, and Ph. Oechslin. SMART: A Many-to-Many Multicast Protocol for ATM. August 1996.

[46] P. Giacomazzi and L. Musumeci. Tranport of IP Controlled-Load Service over ATM Networks. *IEEE Network*, 13(1):36–47, January/February 1999.

[47] S. Giordano and J.-Y. Le Boudec. On Class of Time Varying Shapers with Application to the Renegotiable Varaible Bit Rate Service. April 2000.

[48] S. Golestain. A Self-Clock Fair Queueing Scheme for Broadband Applications. *in Proceedings of IEEE Infocom '94*, June 1994.

[49] S. Golestani. A Stop-and-Go Queueing Framework for Congestion Management. *in Proceedings of ACM SIGCOMM'90*, pages 8–18, Philadelphia, PA, September 1990.

[50] N. Golime. The NIST ATM/HFC Network Simulator, Operation and Programming Guide. In *NISTIR 5703R2*, December 1998.

[51] N. Golmie, Y. Chang, and D. Su. NIST ER Switch Mechanism (an Example). *ATM Forum/95-0695*, June 1995.

[52] M. Grossglauser and K. K. Ramakrishnan. SEAM: Scalable and Efficient ATM Multicast. *Proceedings of INFOCOM'97*, April 1997.

[53] X. Gu, K. Sohraby, and D. R. Vaman. *Control and Performance in Packet, Circuit and ATM Networks.* Kluwer Academic, Boston, 1995.

[54] A. Gupta and R. Rom. Improving VC set-up, management and use via bunching. 1997.

[55] M. Hassan, J. Breen, and M. Atiquzzaman. HG-RCP: A rate-base flow control mechanis for intranets interconnected by ATM networks. *Computer Networks*, 31(22):2361–2379, 1999.

[56] J. Heinanen. Multipoint-to-point VCs. *ATM Forum / 97-0261*, May 1997.

[57] M. G. Hluchyj and N. Yin. On the Queueing Behavior of Multiplexed Leaky Bucket Regulated Sources. *in Proceedings of IEEE INFOCOM '93*, pages 672–679, San Francisco, CA, March 1993.

[58] R. Jain. Congestion Control and Traffic Management in ATM Network: Recent Advances and Survey. *Department of Computer and Information Science, The Ohio State University*, Columbus, August 1995.

[59] R. Jain. Congestion Control and Traffic Management in ATM Networks: Recent Advances and Survey. *Department of Computer and Information Science, The Ohio State University*, Columbus, August 3 1995.

[60] R. Jain and et. al. ERICA switch algorithm: A complete description. *ATM Forum/96-1172*, August 1996.

[61] R. Jain, S. Fahmy, S. Kalyanaraman, and R. Goyal. ABR switch algorithm testing: A case study with ERICA. In *ATM Forum/96-1267*, Octobre 1996.

[62] Raj Jain, Shiv Kalyanaraman, and Ram Viswanathan. The OSU scheme for congestion avoidance in ATM networks: Lessons learnt and extensions. *Performance Evaluation*, 31(1–2):67–88, May 1996.

[63] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. An Efficient Rate Allocatin Algorithm for ATM Networks Providign max-min fairness. In *Technical Report, Computer Engineering Dept., University of California, Santa Cruz*, June 1995.

[64] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Dynamics of an Explicit Rate Allocation Algorithm for Available Bit-Rate (ABR) Service in ATM Net-

works. In *Technical Report, Computer Engineering Dept., University of California, Santa Cruz*, December 1995.

[65] L. Kalampukas. An Eficient Rate Allocation Algorithm for ATM Networks. Master's thesis, University of California, Santa Cruz, June 1995.

[66] C. Kalmanek, H. Kanakia, and S. Keshav. Rate Controlled Servers for Very High-Speed Networks. *in Proceedings of IEEE Global Telecommunications Conference*, pages 300.3.1–300.3.9, San Diego, California, December 1990.

[67] Shivkumar Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, and Bobby Vandalore. The ERICA switch algorithm for ABR traffic management in ATM networks. Technical report, The Ohio State University, Department of Computer and Information Science, Columbus, OH 43210-1277, USA, November 1997.

[68] A. Kantawala, G. Parulkar, J. DeHart, and T. Marz. Supporting DIS Applications using ATM Multipoint Connection Caching. *Proceedings of INFOCOM '97*, Kobe, Japan, April 7–11, 1997.

[69] Y. Katsube, K. Nagami, and H. Esaki. Toshiba's Router Architecture Extensions for ATM: Overview. *RFC 2098*, February 1997.

[70] S. Keshav. *An Engineering Approach to Computer Networking.* Addison Wesley Pub Co, first edition, 1997.

[71] D. Knuth. *The Art of Computer Programming. Volume 3: Sorting and searching.* Addison-Wesley, 1975.

[72] V. G. Kulkarni and N. Gautam. Leaky Buckets: Sizing and Admission Control. March 1996.

[73] K. Lan and J. Heidemann. Multi-scale Validation of Structural Models of Audio Traffic. *Technical Report ISI-TR-544a*, Information Sciences Institute, University of Southern California, Mariana Del Rey, CA, April 2002.

[74] M. Laubach. Classical IP and ARP over ATM. In *Internet Request for Comments 1577*, January 1994.

[75] J. Liebeherr and E. Yilmaz. Workconserving vs. Non-workconserving Packet Scheduling: An Issue Revisited. Technical report, Computer Science Report No. CS-99-05, Department of Computer Science, University of Virginia, at Charlottesville, VA 22903, May 1999.

[76] J. Luciani, D. Katz, D. Piscitello, B. Cole, and N. Doraswamy. NBMA Next Hop Resolution Protocol (NHRP). *RFC 2332*, April 1998.

[77] M. Maher. ATM Signalling Support for IP over ATM - UNI Signalling 4.0 Update. *RFC 2331*, April 1998.

[78] D. J. Mitzel, D. Estrin, S. Shenker, and L. Zhang. A Study of Reservation Dynamics in Integrated Services Packet. *Proceedings of Infocom*, 1996.

[79] W. Melody Moh. On Multicast ABR Protocols for Wireless ATM Networks. In *Proceedings of the IEEE Int. Conference on Network Protocols (ICNP), pp. 24-31, Atlanta, GA*, October 1997.

[80] W. Melody Moh, Y. Chen, and B. Niizowa. Branch-Point Algorithms for Multicast ATM ABR Protocols. In *Proceedings of the IEEE ATM'98 Workshop, pp. 202-211, Fairfax, VA.*,, May 1998.

[81] F. Moreira. Controlo de Tráfego em Redes ATM. Master's thesis, Faculdade de Engenharia da Universidade do Porto, Janeiro 1997.

[82] F. Moreira and J. Ruela. Controlo de Tráfego ABR em Redes ATM. In *Actas da I Conferência Nacional de Telecomunicações*, pages 23–26, Aveiro, Abril 1997. Instituto de Telecomunicações.

[83] F. Moreira and J. Ruela. Algoritmo de Controlo de Tráfego ABR e de Consolidação de Células de Gestão de Recursos para Serviços Multiponto em Redes ATM. In *Actas da 3 Conferência sobre Redes de Computadores*, Viseu, Novembro 2000.

[84] F. Moreira and J. Ruela. A Method for Traffic Scheduling Based on Token Bucket QoS. In *Proceedings of 3rd Conference on Telecomunications, ConfTele2001*, pages 111–115, Figueira da Foz, Portugal, April 23-24, 2001.

[85] P. Narvaez and K.-Y. Siu. An Acknowledgment Bucket Scheme for Regulating TCP Flow over ATM. *in Proceedings of IEEE Globcom*, 31(22):2361–2379, November 1997.

[86] P. Newman and G. Marshall. Backward Explicit Congestion Notification for ATM Local Area Networks. In *Proceedings IEEE GLOBECOM '93, pp. 719-723*, December 1993.

[87] P. Newman and G. Marshall. BECN Congestion Control. In *ATM Forum 94-789R1*, July 1994.

[88] P. Newman, G. Minshall, T. Lyon, and L. Huston. IP Switching and Gigabit Routers. *IEEE Communication Magazine*, 35(1):64–69, January 1997.

[89] J. K-Y. Ng, S. Song, C. Li, and W. Zhao. A New Method for Integrated End-to-End Delay Analysis in ATM Networks. *Journal of Communications and Networks*, June 1999.

[90] T. S. Eugene Ng, D. C. Stephens, I. Stoica, and H. Zhang. Supporting Best-Effort Traffic with Fair Service Curve. *in Proceedings of GLOBCOM'99*, pages 1799–1807, Rio de Janeiro, Brasil, December 1999.

[91] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control - The Single Node Case. *ACM/IEEE Transactions on Networking*, 1(3):344–357, June 1993.

[92] B. V. Patel and C. C. Bisdikian. End-Station Performance under Leaky Buckets Traffic Shaping. *IEEE Network Magazine*, 10(5):40–47, September/October 1996.

[93] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. *rfc3168*, September 2001.

[94] W. Ren. Congestion Control for Data Traffic over ATM Networks. PhD proposal. Available through W. Ren's home page, August 1996.

[95] W. Ren, K. Y. Siu, and H. Suzuki. On the Performance of Congestion Control Algorithms for Multicast ABR Service in ATM. *In Proceedings of IEEE ATM'96 Workshop*, San Francisco, August 1996.

[96] L. Roberts. Enhanced PRCA (Proportional Rate-Control Algorithm). In *ATM Forum 94-0735R1*, August 1994.

[97] L. Roberts. Rate based algorithms for point to multipoint abr Service. *ATM Forum/94-0772R1*, November 1994.

[98] L. Roberts. Proposal to Select a MCR Option. In *ATM Forum 94-0975*, 1995.

[99] E. C. Rosen, A. Viswanathan, and R. Callon. A Proposed Architecture for MPLS. *Internet Draft*, draft-ietf-mpls-arch-00, August 1997.

[100] S. Rosenberg, M. Aissaoui, K. Galway, and N. Giroux. Functionality at the Edge: Designing Scalable Multiservice ATM Networks. *IEEE Communications Magazine*, 3(5):88–99, May 1998.

[101] D. Saha, S. Mukherjee, and S. Tripath. Multi-rate Traffic Shaping and End-to-End Performance Guarantees in ATM Networks. *in Proceedings of 1994 International Conference on Network Protocols (ICNP'94)*, Boston, MA, October 1994.

[102] H. Sariowan, R. L. Cruz, and G. C. Polyzos. Scheduling for Quality of Service Guarantees via Service Curves. *in Proceedings of ICCCN'95*, pages 512–520, September 20-23 1995.

[103] S. Sheinker, C. Partridge, and R. Guarin. General Characterization Parameters for Integrated Services Network Eelements. *RFC 2215*, September 1997.

[104] S. Shenker, C. Partridge, and R. Guarin. Specification of Guaranteed Quality of Service. *RFC 2212*, September 1997.

[105] R. J. Simcoe. Test Configurations for Fairness and others Tests. *ATM Forum/AF-TM 94-0557*, 1994.

[106] Kai-Yeung Siu and Hong-Yi Tzeng. Congestion Control for Multicast Service in ATM Netweorks. *Proceedings of the IEEE GLOBECOM*, 1:310–314, 1995.

[107] Kai-Yeung Siu and Hong-Yi Tzeng. Intelligent Congestion Control for ABR Service in ATM Networks. In *Dept. of Electrical and Computer Engineering, University of California, Irvine*, 1995.

[108] Inc Stardust.com. White Paper - QoS protocols & architectures. *http://www.qosforum.com/white_papers/*, July 1999.

[109] Inc Stardust.com. White Paper - The Need for QoS. *http://www.qosforum.com/white_papers/*, July 1999.

[110] D. Stephens, J. Bennett, and H. Zhang. Implementing Scheduling Algorithms in High Speed Networks. *IEEE JSAC Special Issue on High Performance Switches/Routers*, September 1999.

[111] D. C. Stephens and H. Zhang. Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture. *IEEE INFOCOM '98*, pages 282–290, S. Francisco, CA, March 1998.

[112] I. Stoica, H. Zhang, and T. S. Eugene Ng. A Herarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Services. *in Proceedings of SIGCOMM'97*, pages 1103–1111, Cannes, France, 1997.

[113] P. P. Tang and T.-Y. C. Tai. Network Traffic Characterization Using Token Bucket Model. *In Proceedings of the IEEE INFOCOM'99, http://www.ieee-infocom.org/1999/papers/01b$_0$3.pdf, NewYork, May*1999.

[114] J. S. Turner. Extending ATM Networks for Efficiente Reliable Multicast. *Proceedings of Workshop on Communication and Architectural Support for Network-Based Parallel Computing*, Springer Verlag, February 1997.

[115] S. Valaee. A Recursive Estimator of Worst-Case Burstiness. *IEEE/ACM Tansactions on Networking*, 9(2):211–222, April 2001.

[116] B. Vandalore, R. Jain, R. Goyal, and S. Fahmy. Design and analysis of queue control functions for explicit rate switch schemes. In *Proceedings of IC3N '98, Lafayette, LA, pp. 780-786*, October 1998.

[117] D. Varma, H. Zhang, and D. Ferrari. Guaranteeing Delay Jitter Bounds in Packet Switching Networks. *in Proceedings of Tricomm'91*, pages 35–46, Chapel Hill, North Carolina, April 1989.

[118] A. Viswanathan, N. Feldman, R. Boivie, and R. Woundy. ARIS: Aggregate Route-Based IP Switching. *Internet-Draft, draft-viswanathan-aris-overview-00*, March 1997.

[119] D. G. Waddington and F. Chang. Realizing the transition to IPv6. *IEEE Communications Magazine*, 40(6):138–148, June 2002.

[120] N. Yin and M. Hluchyji. On-Closed-Loop Rate-Controls for ATM Cell Relay Networks. In *Proceedings IEEE INFOCOM '94*, June 1994.

[121] H. Zhang. Providing End-to-End Performance Guarantees Using Non-working-conserving Disciplines. *in Computer Communications: Special Issue on System Support for Multimedia Computing*, 18(10), October 1995.

[122] H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks. *in IEEE Proceedings 1995*, 83(10):1374–1399, October 1995.

[123] H. Zhang and D. Ferrari. Rate-Controlled Static-Priority Queueing. *in Proceedings of IEEE Infocom '93*, pages 227–236, April 1993.

[124] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks. *in Proceedings of ACM SIGCOMM'90*, September 1990.

[125] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network Magazine*, pages 8–18, September 1993.

Apêndice A

Network Calculus

O anexo está dividido em duas secções, na primeira é apresentada detalhadamente a ferramenta matemática *Network Calculus*, enquanto que, na segunda, a ferramenta é aplicada ao modelo desenvolvido.

A.1 Ferramenta matemática

Esta ferramenta matemática pode ser vista como a aplicação da álgebra *Max-Min plus* a problemas relacionados com fluxos; uma explicação mais detalhada é apresentada no livro publicado por Le Boudec [12].

A.1.1 Operações

No *Network Calculus* é considerada a álgebra *max-min plus*, em que as operações são alteradas, relativamente à algebra convencional, da seguinte forma: a adição converte-se na operação de obtenção do mínimo e a multiplicação na operação de adição, resultando na estrutura algébrica $(\mathcal{R} \cup \{+\infty\}, \wedge, +)$, com $\wedge$ a representar a operação de obtenção do ínfimo ou se existir, o mínimo, tal que $a \wedge b = min\{a, b\}$ [12]; por exemplo, a propriedade de distribuição da adição em relação à multiplicação na álgebra convencional é

$$(2 + 6) \times 3 = (2 \times 3) + (6 \times 3) = 6 + 18 = 24$$

enquanto que na álgebra *min-plus* é

$$(2 \wedge 6) + 3 = (2 + 3) \wedge (6 + 3) = 5 \wedge 8 = 5.$$

Min-plus convolution

Uma função f é crescente em sentido lato se e só se $f(s) \leq f(t)$ para todo $s \leq t$. É assumido que $\mathcal{F}$ representa um conjunto de sequências ou funções crescentes em sentido lato tal que $f(t) = 0$ para $t < 0$, sendo $\mathcal{R}^+ = [\ 0, +\infty]$ a gama de variação do conjunto $\mathcal{F}$.

Esta nova estrutura algébrica, $(\mathcal{R} \cup \{+\infty\}, \wedge, +)$, em conjunto com a operação convencional de convolução

$$(f \otimes g)(t) = \int_{-\infty}^{+\infty} f(t-s)g(s)ds$$

torna-se, quando $f(t)$ e $g(t)$ são duas funções com valor zero para $t < 0$, em

$$(f \otimes g)(t) = \int_{o}^{t} f(t-s)g(s)ds.$$

e permite definir uma extensão da definição anterior da seguinte forma:

Definição 2 *(Min-plus convolution [12]) Sejam f e g duas funções de $\mathcal{F}$. A convolução min-plus de f e g é a função*

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{f(t-s) + g(s)\},$$

se $t < 0$, $(f \otimes g)(t) = 0$.

Min-plus deconvolution

A operação dual da operação anterior é a *min-plus deconvolution*; no entanto é alterada a operação de obtenção do inferior (ou mínimo) para a obtenção do superior, isto é, a notação $\vee$ representa o superior ou, se existir, o máximo, tal que $a \vee b = max\{a, b\}$.

Definição 3 *(Min-plus deconvolution [12]) Sejam f e g duas funções ou sequências de $\mathcal{F}$. A min-plus deconvolution de f e g é a função*

$$(f \oslash g)(t) = \sup_{u \geq 0} \{f(t+u) - g(u)\}.$$

Contrariamente à operação anterior, a função $(f \oslash g)(t)$ não é necessariamente zero se $t < 0$.

Tabela A.1: Funções de referência

Rate-latency	$\beta_{R,T} = R[t-T]^+ = \begin{cases} R(t-T), & \text{se } t > T \\ 0, & t \leq T \end{cases}$
Peak rate	$\lambda_R(t) = \begin{cases} Rt, & \text{se } t > 0 \\ 0, & t \leq 0 \end{cases}$
Affine	$\gamma_{r,b}(t) = \begin{cases} rt + b, & \text{se } t > 0 \\ 0, & t \leq 0 \end{cases}$

A.1.2 Funções de referência

As funções de referência utilizadas para a modelação dos resultados obtidos são a função *Rate-latency* ($\beta_{R,T}$), *Peak rate* (λ_R) e a função *Affine* ($\gamma_{r,b}$); as funções encontram-se na Tabela A.1 e graficamente representadas na Figura A.1.

Antes de apresentar as definições das *curvas de chegada* e de *serviço* é necessário indicar o significado das funções utilizadas. Assim, considerando um sistema S (que pode representar, por exemplo, um *buffer* a ser servido a um débito constante, um nó de comunicações, etc.), que recebe, processa e envia tráfego. O tráfego de entrada é descrito pela função $A^{in}(t)$ que representa o tráfego acumulado, enquanto que a função $B^{out}(t)$ representa o tráfego acumulado de saída do sistema.

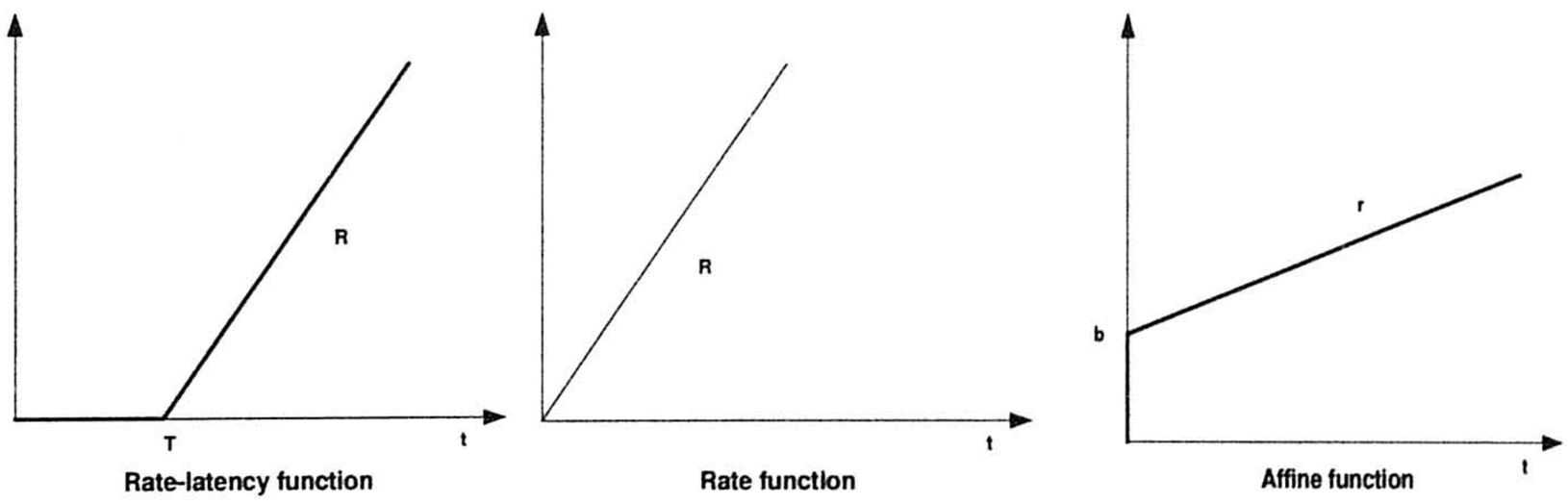

Figura A.1: Funções de referência

A.1.3 Curvas de Chegada

Nesta subsecção serão definidas as *curvas de chegada* [102, 27, 12] que serviram para modelar matematicamente o tráfego que chega a um determinado nó da rede.

Definição 4 *(Curva de Chegada [12]) Dada uma função crescente $\alpha(t)$ definida para $t \geq 0$, é possível dizer que um fluxo A^{in} é limitado por $\alpha(t)$ se e só se para todo $s \leq t$: $A^{in}(t) - A^{in}(s) \leq \alpha(t-s)$. É possível dizer que A^{in} tem α como uma curva de chegada, ou também que A^{in} é α-smooth.*

Definição 5 *(Burstiness Constraints [102, 27]) Dada uma função $b(.)$ não decrescente, designada por curva de chegada, é possível dizer que o tráfego de entrada A^{in} é b-smooth se $A^{in}[s] \leq b(t-s)$ para todo o s e t que satisfaçam a condição $s \leq t$. No caso especial em que $b(.)$ é uma função affine, $b(x) = \sigma + \rho x$, A^{in} é (σ, ρ)-smooth.*

Relacionado com este tipo de curvas foi introduzido o conceito de *Sub-additivity closure* que indica o seguinte: dada uma função $f \in \mathcal{F}$, se $f(0) = 0$, então $f \geq f \otimes f \geq 0$. Repetindo esta operação, obter-se-á uma sequência de funções que são cada vez mais "pequenas" e convergem para uma função limite que é a maior função *sub-additivity* menor do que f e com valor zero em $t = 0$, sendo a definição formal

Definição 6 *(Sub-additivity closure [12]) Assume-se que f é uma função de $\mathcal{F}$. Designa-se por $f^{(n)}$ a função obtida através da repetição de $(n-1)$ convoluções de f com ela própria. Por convenção, $f^{(0)} = \delta_0$, $f^{(1)} = f$, $f^{(2)} = f \otimes f$, etc. Então a sub-additive closure de f, designada por $\bar{f}$, é definida por*

$$\bar{f} = \delta_0 \wedge f \wedge f \otimes f \wedge f \otimes f \otimes f \wedge ... = \inf_{n \geq 0} \{f^{(n)}\}.$$

Entre todas as funções *sub-additivity* que são menores do que f e com valor zero em $t = 0$, existe uma que é o limite superior para todas as outras; ela é igual a $\bar{f}$, como definido no seguinte teorema

Teorema 7 *(Sub-additivity closure [12]) Seja f uma função ou uma sequência de $\mathcal{F}$, e sendo $\bar{f}$ a sua sub-additivity closure, então*

(i) $\bar{f} \leq f$, $f \in \mathcal{F}$ e f é sub-additive.

(ii) se a função $g \in \mathcal{F}$ é sub-additive, com $g(0) = 0$ e $g \leq f$, então $g \leq \bar{f}$.

É possível afirmar, com base no teorema anterior, que a *sub-additivity closure* de uma função α é $\bar{\alpha}$ tal que $\bar{\alpha} \leq \alpha$.

Definição 8 *([12]) Considere-se uma função $\alpha \in \mathcal{F}$. É possível dizer que α é uma "boa" função se e só se uma das seguintes propriedades se verificar:*

(1) α é sub-additive e $\alpha(0) = 0$;

(2) $\alpha = \alpha \otimes \alpha$;

(3) $\alpha \oslash \alpha = \alpha$ e

(4) $\alpha = \bar{\alpha}$.

Teorema 9 *(Reduction of Arrival Curve to a Sub-Additivity One [12]). Dizer que um fluxo é constrangido por uma função crescente α é equivalente a dizer que é constrangido pela sua sub-additivity closure $\bar{\alpha}$.*

Lema 10 *Um fluxo A^{in} é constrangido pela curva de chegada α se e só se $A^{in} \leq A^{in} \otimes \alpha$.*

A.1.4 Curvas de Serviço

As *curvas de serviço* apresentadas em [102, 27, 12] têm como função garantir os requisitos negociados inicialmente nos nós da rede e definem-se da seguinte forma:

Definição 11 *(Curva de Serviço [12]) Considerando um sistema S e um fluxo a atravessar S com funções de entrada e saída designadas por A^{in} e B^{out}, é possível dizer que S oferece a um fluxo uma curva de serviço β se e só se $\beta \in \mathcal{F}$ e $B^{out} \geq A^{in} \otimes \beta$.*

Para ilustrar a definição a Figura A.2 é um bom exemplo. Esta definição significa que β é uma função crescente com $\beta(0) = 0$, e que para todo o $t \geq 0$,

$$B^{out}(t) \geq \inf_{s \leq t}(A^{in}(s) + \beta(t - s)).$$

Contudo, mesmo que o sistema apresente uma ocupação máxima a *curva de serviço* definida anteriormente é sempre garantida como estabelecido pela seguinte definição

Definição 12 *(Strict Service Curve [12]). É possível afirmar que um sistema S oferece uma strict service curve β a um fluxo se durante qualquer período "backlogged" de duração u, o fluxo de saída é pelo menos igual a $\beta(u)$.*

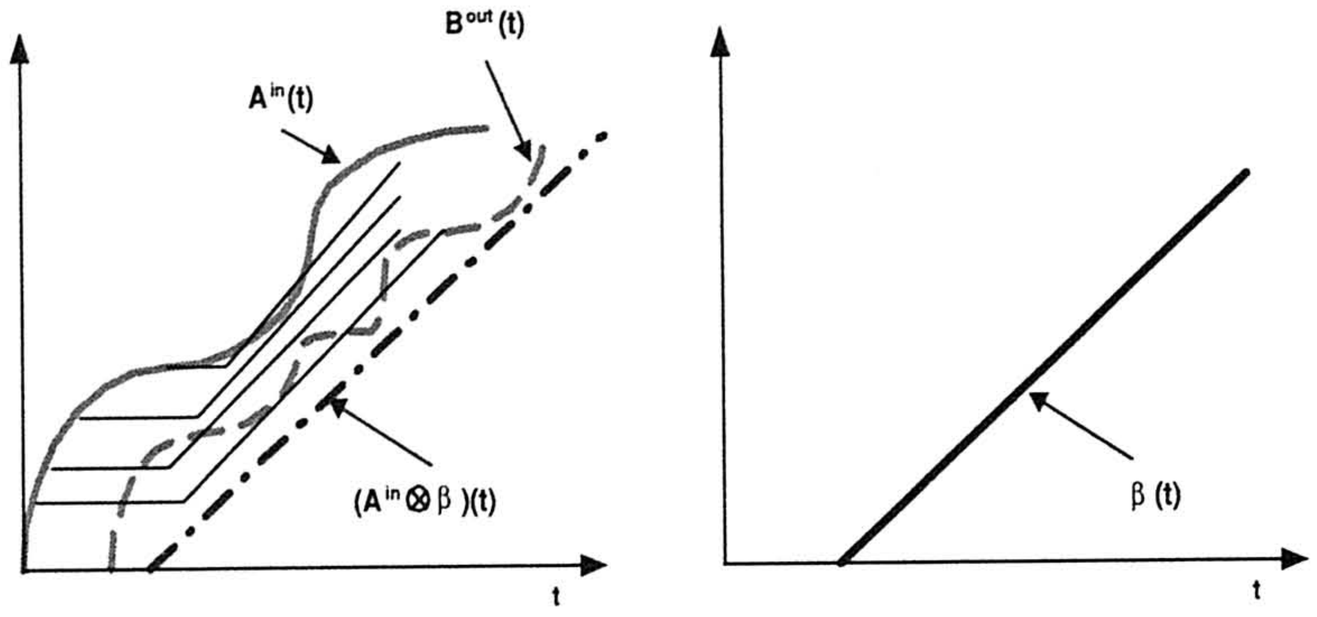

Figura A.2: Definição da curva de serviço. A função B^{out} deve estar acima de $A^{in} \otimes \beta$, que é a menor das envolventes de todas as curvas $A^{in}(t_0) + \beta(t - t_0)$.

A.1.5 Limites

Em [27] e [12] são estabelecidas as *curvas de chegada e de serviço* para obter os limites de atraso e de *backlog*.

Inicialmente são apresentados os teoremas de L. Cruz relativamente a limites, isto é, para a obtenção dos atrasos é estabelecido o seguinte teorema

Teorema 13 *(Upper Bound on Delay [27]) Considerando um servidor que garante a curva de serviço S(.) para uma conexão e supondo que o tráfego de entrada A^{in} é b-smooth, então para todo o t, o atraso $D[t]$ é limitado superiormente por*

$$D[t] = \max_{s:s\geq 1} \min\{\Delta : \Delta \geq 0 \ e \ b(s) \leq S(s+\Delta)\}$$

e para a obtenção do limite máximo de ocupação dos *buffers* é definido o teorema seguinte

Teorema 14 *(Upper Bound on Backlog [27]) Considerando um servidor que garante a curva de serviço $S(.)$ para uma conexão e supondo que o tráfego de entrada A^{in} é b-smooth, então para todo o t, a ocupação $B^*[t]$ é limitada superiormente por*

$$B^*[t] \leq \max_{s:s\geq 0}\{b(s) - S(s)\}.$$

De forma a uniformizar o significado das funções utilizadas é necessário referir que $x(t)$ e $y(t)$ têm o mesmo significado que $A^{in}(t)$ e $B^{out}(t)$, respectivamente.

Os limites definidos por Le Boudec são representados pelos seguintes teoremas. O primeiro teorema mostra que a ocupação é limitada pela máxima distância vertical entre as *curvas de chegada* e de *serviço*, isto é,

Teorema 15 *(Backlog Bound [12, 11, 13]) Considere-se um fluxo constrangido pela curva de chegada α, e que atravessa um sistema que oferece uma curva de serviço β. A ocupação $x(t) - y(t)$ para todo o t satisfaz*

$$x(t) - y(t) \leq \sup_{s \geq 0} \{\alpha(s) - \beta(s)\}$$

enquanto que a distância horizontal pode ser dada por $\Delta(s) = inf\{\tau \geq 0 : \alpha(s) \leq \beta(s+\tau)\}$ em que $\Delta(s)$ é o atraso para um sistema hipotético com uma *curva de chegada* α(s) e uma *curva de serviço* $\beta(s)(\alpha(s) \leq \beta(s))$. Assumindo que $h(\alpha, \beta)$ é o supremo de todos os valores de $\Delta(s)$, o limite do atraso é dado pelo seguinte teorema

Teorema 16 *(Delay Bound [12, 11, 13]) Considere-se um fluxo constrangido por uma curva de chegada α, e que atravessa um sistema que oferece uma curva de serviço β. O atraso $d(t)$ para todo o t satisfaz a condição $d(t) \leq h(\alpha, \beta)$.*

Teorema 17 *(Output Flow [12, 11, 13]) Assumindo um fluxo constrangido por uma curva de chegada α, e que atravessa um sistema que oferece uma curva de serviço β, a curva de saída é constrangida pela curva de chegada $\alpha^* = \alpha \oslash \beta$.*

A.2 Justificação formal do modelo

Para a justificação formal do modelo proposto recorreu-se ao *Network Calculus* [12], como apresentado no Capítulo 3 e na secção anterior.

Inicialmente serão realizadas extensões dos limites do atraso e do *backlog* com o objectivo de criar novas definições e teoremas que permitam relacionar os parâmetros do modelo com as definições já existentes. Após estas definições serão utilizadas as técnicas desenvolvidas por Le Boudec para verificar formalmente se os limites admitidos como pressupostos são os mais adequados, bem como a representação matemática da curva de saída (débito atribuído a cada fluxo), com base nas *Curvas de Chegada* e de *Serviço* escolhidas previamente.

A.2.1 Definição de novos limites

Os parâmetros definidos no modelo (X, Z, Q) podem ser relacionados com as *curvas de entrada* e de serviço e pelos teoremas e definições estabelecidos no *Network Calculus* relativamente à definição de limites de atraso e de *backlog*.

O Teorema 15 permite determinar o limite máximo do *backlog* através da expressão

$$x(t) - y(t) \leq \sup_{s \geq 0}\{\alpha(s) - \beta(s)\}$$

que significa que no instante t é obtido o valor máximo (ou supremo) da diferença entre as *Curvas de Chegada* e de *Serviço*, definidas anteriormente.

A ideia de garantir um determinado débito a uma fonte implica que a condição $y(t) - \beta(t) \geq 0$ se verifique para todo o t. Contudo, é possível explorar ainda mais esta relação, uma vez que ela fornece uma informação adicional que vai ser de extrema importância no modelo desenvolvido, nomeadamente ter sempre presente a quantidade de largura de banda que uma determinada fonte está a beneficiar acima do débito garantido para que seja possível beneficiar igualmente as suas concorrentes.

Definição 18 *(Débito distribuído em excesso em relação ao garantido) Assumindo que X é uma função que verifica $X(0) = 0$, é possível afirmar que um servidor garante a curva de serviço $\beta(t)$ para um fluxo se para qualquer t tal que $y(t) \geq \beta(t)$ o débito distribuído em excesso é dado por $X(t) = y(t) - \beta(t) \geq 0$.*

Com base na Definição 18 é possível garantir que em termos médios uma fonte é sempre servida pelo menos ao seu débito garantido porque assim que o valor da função X_i é negativa, a fonte i é servida de imediato.

O parâmetro designado por Z tem um papel relevante na tomada de decisão relativamente à fonte que vai ser servida a seguir, uma vez que o parâmetro permite saber quantas células de um determinado *burst* já entraram no ED e foram processadas pelo TB associado. Este parâmetro tem uma interferência directa sempre que uma fonte inicia a sua actividade, e no instante em que se verifica que uma segunda fonte iniciou a sua actividade após ter sido já processado mais de metade das células do *burst* da primeira; a ilustração gráfica destas situações pode ser encontrada na secção 5.5 Cenário 2. Com base neste pressuposto é possível estabelecer uma nova definição relativamente à evolução dos *bursts* devido ao papel que esta informação vai desempenhar.

Definição 19 *(Evolução do Burst) Dada uma função $\alpha(t)$ não decrescente, designada por curva de chegada, é possível dizer que o tráfego de entrada $x(t)$ é constrangido por $\alpha(t)$ se $x(t) \leq \alpha(t)$ para todo o $t > 0$. No caso especial em que α é uma função affine, $\alpha(t) = \gamma_{r,b}$, $x(t)$ é (r, b)-smooth e dada uma função $Z(t)$ com $Z(0) = b$ é possível afirmar que o estado do burst no instante t é $Z(t) = \alpha(t) - x(t) \geq 0$.*

Com base no paralelismo das curvas é possível verificar que o limite máximo de ocupação nunca será superior ao tamanho máximo do *bucket*, b, logo $Q(t) \leq b$, o que permite redefinir o Teorema 15 da seguinte forma

Teorema 20 *(Limite de Backlog) Dado um fluxo que é constrangido pela curva de chegada $\alpha = \gamma_{r,b}$ e atravessa um sistema que oferece uma curva de serviço $\beta = \lambda_r$, o backlog $Q(t) = x(t) - y(t)$ para todo o t satisfaz a condição $x(t) - y(t) \leq b$ logo o limite máximo será*

$$b = sup_{s \geq 0}\{\alpha(s) - \beta(s)\},$$

mas como as rectas são paralelas $b = \alpha(s) - \beta(s)$ com $Q(t) \leq b$.

Prova. Para provar o Teorema 20 basta substituir as expressões das Definições 18 e 19 e verificar a condição $x(t) - y(t) \leq b$, isto é, com base nos pressupostos iniciais

$b = X(t) + Z(t) + Q(t)$ com $X(t) = y(t) - \beta(t)$, $Z(t) = \alpha(t) - x(t)$ e $Q(t) = x(t) - y(t)$

logo

$$b = y(t) - \beta(t) + \alpha(t) - x(t) + x(t) - y(t)$$

$$b = \alpha(t) - \beta(t).$$

■

A.2.2 *Curva de Chegada*

A *Curva de Chegada* escolhida para este método foi a curva conhecida por *Affine Arrival Curve* $\gamma_{r,b}$ que está relacionada com os TBs que são a principal base do modelo desenvolvido.

Uma vez que foi assumido que o modelo desenvolvido se baseia na utilização de TBs e a função de referência que os representa é $\gamma_{r,b}$ é necessário verificar, segundo a

Definição 8, que esta é uma "boa" função. Para verificar se a função é "boa" basta confirmar uma das suas propriedades; contudo vão ser verificadas as quatro:

1. A função tem propriedades de sub-aditividade se e só se $f(s+t) \leq f(s) + f(t)$ e $f(0) = 0$

$$\begin{aligned} \gamma_{r,b}(s+t) &\leq \gamma_{r,b}(t) + \gamma_{r,b}(s) \\ b + r(s+t) &\leq b + rt + b + rs \\ b + rt + rs &\leq b + rt + b + rs \\ b &\leq 2b \end{aligned}$$

$\gamma_{r,b}(0) = 0$, como definido aquando da apresentação das funções de referência, por isso, bastaria esta prova para verificar que a função tem a propriedade de sub-aditividade;

2. Segunda propriedade ($\alpha = \alpha \otimes \alpha$),

$$\begin{aligned} (\gamma_{r,b} \otimes \gamma_{r,b})(t) &= \inf_{0 \leq s \leq t} \{\gamma_{r,b}(t-s) + \gamma_{r,b}(s)\} \\ &= \inf_{s=0} \{\gamma_{r,b}(t-s) + \gamma_{r,b}(s)\} \wedge inf_{0<s\leq t} \{\gamma_{r,b}(t-s) + \gamma_{r,b}(s)\} \\ &= \{b + rt\} \wedge inf_{0<s\leq t} \{b + r(t-s) + b + rs\} \\ &= \{b + rt\} \wedge inf_{0<s\leq t} \{b + rt - rs + b + rs\} \\ &= \{b + rt\} \wedge \{2b + rt\} \\ &= \{b + rt\} \end{aligned}$$

fica assim provada a segunda propriedade porque $\gamma_{r,b} = \gamma_{r,b} \otimes \gamma_{r,b}$, pelo que se pode concluir que a propriedade $\alpha = \bar{\alpha}$ (quarta propriedade) também se verifica porque $\gamma_{r,b} = \bar{\gamma}_{r,b}$.

3. A terceira propriedade indica que $\alpha \oslash \alpha = \alpha$, logo é necessário verificar para $\gamma_{r,b} \oslash \gamma_{r,b} = \gamma_{r,b}$

$$
\begin{aligned}
(\gamma_{r,b} \oslash \gamma_{r,b})(t) &= \sup_{u \geq 0}\{\gamma_{r,b}(t+u) - \gamma_{r,b}(u)\} \\
&= \sup_{u=0}\{\gamma_{r,b}(t+u) - \gamma_{r,b}(u)\} \vee \sup_{u>0}\{\gamma_{r,b}(t+u) - \gamma_{r,b}(u)\} \\
&= \sup_{u=0}\{\gamma_{r,b}(t+u)\} \vee \sup_{u>0}\{b + rt + ru - b - ru\} \\
&= \sup_{u=0}\{b + rt + ru\} \vee \sup_{u>0}\{rt\} \\
&= \{b + rt\} \vee \{rt\} \\
&= \{b + rt\} \\
&= \gamma_{r,b}
\end{aligned}
$$

Aplicando a este modelo é possível, através da convolução entre a função λ_R ($\lambda_R = R^{in} \times t$) com a *Curva de Chegada* $\gamma_{r,b}$, com o valor de r igual ao débito médio garantido (SCR) e o valor do tamanho do *bucket* (b), verificar que

$$
A^{in} \leq A^{in} \otimes \alpha \Longleftrightarrow \lambda_R \leq \lambda_R \otimes \gamma_{r,b} \Longleftrightarrow (\lambda_R \otimes \gamma_{r,b})(t) = \inf_{0 \leq s \leq t}\{\lambda_R(t-s) + \gamma_{r,b}(s)\}.
$$

Considerando a definição

$$
(\lambda_R \otimes \gamma_{r,b})(t) = \inf_{0 < s < t}\{\lambda_R(t-s) + \gamma_{r,b}(s)\}
$$

é possível fazer a respectiva verificação para $0 \leq s \leq t$, sabendo que $R > r$, a curva resultante é

$$
\begin{aligned}
(\lambda_R \otimes \gamma_{r,b})(t) &= \inf_{0 \leq s \leq t}\{\lambda_R(t-s) + \gamma_{r,b}(s)\} \\
&= inf_{0 < s \leq t}\{R(t-s) + b + rs\} \wedge \inf_{s=0}\{\lambda_R(t-s) + \gamma_{r,b}(s)\} \\
&= \inf_{0 \leq s \leq t}\{Rt - Rs + b + rs\} \wedge \{\lambda_R(t-0) + \gamma_{r,b}(0)\} \\
&= \inf_{0 \leq s \leq t}\{Rt + b + (r-R)s\} \wedge \{\lambda_R(t) + 0\} \\
&= \{Rt + b + (r-R)t\} \wedge \{\lambda_R(t)\} \\
&= \{Rt + b + rt - Rt\} \wedge \{\lambda_R(t)\} \\
&= \{b + rt\} \wedge \{Rt\}
\end{aligned}
$$

o que prova que o fluxo de entrada, A^{in}, é limitado (ou constrangido) pela função $\gamma_{r,b}$.

A.2.3 *Curva de Serviço*

Relativamente à *Curva de Serviço* $\beta_{r,T}$ (com $T = 0$) é possível obter a curva de referência λ_r, uma vez que existe uma garantia de serviço de pelo menos o débito médio que as fontes deste tipo necessitam.

A função escolhida para representar a *Curva de Serviço* foi λ_r e por isso é necessário verificar, recorrendo à Definição 8, se esta é uma "boa" função. Para verificar se a função é "boa" basta confirmar uma das propriedades; no entanto vão ser verificadas as quatro:

1. A função tem propriedades de sub-aditividade se e só se $f(s+t) \leq f(s) + f(t)$ e $f(0) = 0$

$$\begin{aligned} \lambda_r(s+t) &\leq \lambda_r(t) + \lambda_r(s) \\ r(s+t) &\leq rt + rs \\ rt + rs &\leq rt + rs \end{aligned}$$

$\lambda_r(0) = 0$, como definido aquando da apresentação das funções de referência, por isso, bastaria esta prova;

2. A segunda propriedade, $\lambda_r = \lambda_r \otimes \lambda_r$,

$$\begin{aligned} (\lambda_r \otimes \lambda_r)(t) &= inf_{0 \leq s \leq t}\{\lambda_r(t-s) + \lambda_r(s)\} \\ &= \inf_{0 \leq s \leq t}\{r(t-s) + r(s)\} \\ &= \inf_{0 \leq s \leq t}\{rt - rs + rs\} \\ &= rt \\ &= \lambda_r \end{aligned}$$

fica assim provada a segunda propriedade porque $\lambda_r = \lambda_r \otimes \lambda_r$, enquanto que a quarta propriedade também se verifica, isto é,

$$f^{(n)} = \inf_{i>0}\{f^{(i)}\},$$

logo

$$\inf_{i \geq 0}\{\lambda_r^{(i)}\} = \lambda_r \wedge \lambda_r \otimes \lambda_r \wedge \lambda_r \otimes \lambda_r \otimes \lambda_r \wedge \ldots = \lambda_r,$$

sendo por isso $\lambda_r = \bar{\lambda}_r$ (fica verificada a quarta propriedade).

3. A terceira propriedade indica que $\alpha \oslash \alpha = \alpha$, logo é necessário verificar se $\lambda_r \oslash \lambda_r = \lambda_r$

$$\begin{aligned}(\lambda_r \oslash \lambda_r)(t) &= \sup_{u\geq 0}\{\lambda_r(t+u) - \lambda_r(u)\}\\ &= \sup_{u\geq 0}\{rt + ru - ru\}\\ &= \sup_{u\geq 0}\{rt\}\\ &= rt\\ &= \lambda_r\end{aligned}$$

Para este modelo a Definição 12 aplica-se directamente porque a *Curva de Serviço* é definida *à priori*, $\beta(t) = r \times t$, logo a curva de saída é

$$B^{out}(t) \geq \inf_{s\leq t}\{A^{in}(s) + \beta(t-s)\}$$

com $A^{in}(t) = \lambda_R$ e $\beta(t) = \lambda_r$, então

$$B^{out}(t) \geq \inf_{s\leq t}\{\lambda_R(s) + \lambda_r(t-s)\}.$$

Considerando a definição

$$(\lambda_R \otimes \lambda_r)(t) = \inf_{s\leq t}\{\lambda_R(s) + \lambda_r(t-s)\}$$

é possível fazer a respectiva verificação para $0 \leq s \leq t$, sabendo que $R > r$, a curva resultante é

$$\begin{aligned}(\lambda_R \otimes \lambda_r)(t) &= \inf_{0\leq s\leq t}\{\lambda_R(t-s) + \lambda_r(s)\}\\ &= \inf_{0\leq s\leq t}\{R(t-s) + r(s)\}\\ &= \inf_{0\leq s\leq t}\{Rt - sR + rs\}\\ &= \inf_{0\leq s\leq t}\{Rt + (r-R)s\}\\ &\text{ pois } (r-R) < 0 \text{ e } t \leq s\\ &= \{Rt + (r-R)t\}\\ &= \{Rt + rt - Rt\}\\ &= \{rt\}\\ &= \lambda_r\end{aligned}$$

o que prova que a *Curva de Serviço*, λ_r, é a definida como pressuposto inicial deste modelo e que é garantida quando o débito de saída do canal tem de ser equitativamente dividido por todos os fluxos que estão a consumir a largura de banda na totalidade; no entanto deve ser sempre verificada a condição

$$B(t)^{out} \geq \sum_{j=1}^{N} \lambda_r^j$$

É ainda necessário testar se o modelo em discussão obedece ao Teorema 17, que indica que existe uma *Curva de Chegada* $\alpha^* = \alpha \oslash \beta$, que não é mais do que a curva que limita a curva de saída obtida. Como $\alpha = \gamma_{r,b}$ e $\beta = \lambda_r$ o resultado da operação de deconvolução é

$$\begin{aligned}
(\gamma_{r,b} \oslash \lambda_r)(t) &= \sup_{u \geq 0}\{\gamma_{r,b}(t+u) - \lambda_r(u)\} \\
&= \sup_{u=0}\{\gamma_{r,b}(t+u) - \lambda_r(u)\} \vee \sup_{u>0}\{\gamma_{r,b}(t+u) - \lambda_r(u)\} \\
&= \sup_{u=0}\{b + rt + ru\} \vee \sup_{u>0}\{b + rt + ru - ru\} \\
&= \{b + rt\} \vee \{b + rt\} \\
&= \{b + rt\} \\
&= \gamma_{r,b}
\end{aligned}$$

verificando-se que a *Curva de Chegada* que limita a curva de saída é a mesma que limita o fluxo de entrada e, como a *curva de saída* é sempre inferior ou igual à *Curva de Chegada* fica justificada a escolha das *Curvas de Chegada* e de *Serviço*.

A.2.4 Definição da *curva de saída*

A obtenção de uma *curva de saída* pode ser conseguida através das ferramentas de modelação descritas. Com base no comportamento desejado relativamente aos débitos obtidos na distribuição da largura de banda pelos fluxos em competição, como observado, por exemplo, na Figura 5.4 (b), é possível obter a mesma representação gráfica através da combinação de algumas funções de referência apresentadas na secção A.1.

A primeira abordagem será realizada através da combinação das funções de referência já definidas, sendo que a primeira situação é verificar o que acontece quando se realiza a convolução de uma função com ela própria, em especial $\beta_{R,T}$, uma vez que um dos objectivos é a distribuição dos atrasos e esta função representa uma curva com um atraso de T. Seguindo a definição de *min-plus convolution*, temos

$$(\beta_{R,T} \otimes \beta_{R,T})(t) = \inf_{0 \leq s \leq t}\{\beta_{R,T}(t-s) + \beta_{R,T}(s)\}$$

para $t \leq T$,

$$(\beta_{R,T} \otimes \beta_{R,T})(t) = \inf_{0 \leq s \leq t}\{\beta_{R,T}(t-s) + \beta_{R,T}(s)\} = 0$$

enquanto que para $t > T$

$$
\begin{aligned}
(\beta_{R,T} \otimes \beta_{R,T})(t) &= \inf_{0\leq s\leq T}\{\beta_{R,T}(t-s)+\beta_{R,T}(s)\} \wedge \inf_{T\leq s<t}\{\beta_{R,T}(t-s)+\beta_{R,T}(s)\} \\
&\wedge inf_{s=t}\{\beta_{R,T}(t-s)+\beta_{R,T}(s)\} \\
&= \inf_{0\leq s\leq T}\{R[t-s-T]^{+}+R[s-T]^{+}\} \wedge \inf_{T\leq s<t}\{R[t-s-T]^{+}+ \\
&R[s-T]^{+}\} \wedge \inf_{s=t}\{R[t-s-T]^{+}+R[s-T]^{+}\} \\
&= \{R(t-2T)+0\} \wedge \{R(t-2T)+0\} \wedge \{0+R(t-T)\} \\
&= \{R(t-2T)\} \wedge \{R(t-2T)\} \wedge \{R(t-T)\} \\
&= \{R(t-2T)\} \wedge \{R(t-T)\} \\
&= \{R(t-2T)\} \\
&= \beta_{R,2T}
\end{aligned}
$$

se existir a necessidade de continuar a realizar a convolução da função resultante com a própria o resultado será

$$(\beta_{R,T} \otimes \beta_{R,2T})(t) = \beta_{R,3T}$$

e assim sucessivamente.

Quando a função a estudar tem um ponto de origem diferente de zero, por exemplo, $k+\beta_{R,T}$ a convolução desta função com ela própria resulta no seguinte

$$
\begin{aligned}
(k+\beta_{R,T} \otimes k+\beta_{R,T})(t) &= (\beta_{R,T} \otimes \beta_{R,T})(t)+2k \\
&= \beta_{R,2T}+2k
\end{aligned}
$$

$$
\begin{aligned}
(2k+\beta_{R,2T} \otimes k+\beta_{R,T})(t) &= (\beta_{R,2T} \otimes \beta_{R,T})(t)+3k \\
&= \beta_{R,3T}+3k
\end{aligned}
$$

e assim sucessivamente. Generalizando a convolução desta para n a expressão resultante é

$$
\begin{aligned}
((n-1)k+\beta_{R,(n-1)T} \otimes k+\beta_{R,T})(t) &= (\beta_{R,(n-1)T} \otimes \beta_{R,T})(t)+nk \\
&= \beta_{R,nT}+nk
\end{aligned}
$$

e a combinação destas expressões para a obtenção da curva mínima é conseguida da seguinte forma

$$
\begin{aligned}
\beta_{R,nT}+nk &= \inf_{i\geq 0}\{ik+\beta_{R,iT}\} \\
&= \lambda_R \wedge \{k+\beta_{R,T}\} \wedge \{2k+\beta_{R,2T}\} \wedge \{3k+\beta_{R,3T}\} \wedge \{4k+\beta_{R,4T}\} \wedge ...
\end{aligned}
$$

com o caso geral dado por

$$\inf_{i\geq 0}\{ik + \beta_{R,iT}\} \tag{A.1}$$

Quando se conhece a curva de saída desejada (ver Figuras 5.4 (b) e 5.5 (b)) e se tenta modelar matematicamente é necessário analisar a curva e, a partir desse momento, com base nas expressões apresentadas anteriormente obter um caso geral.

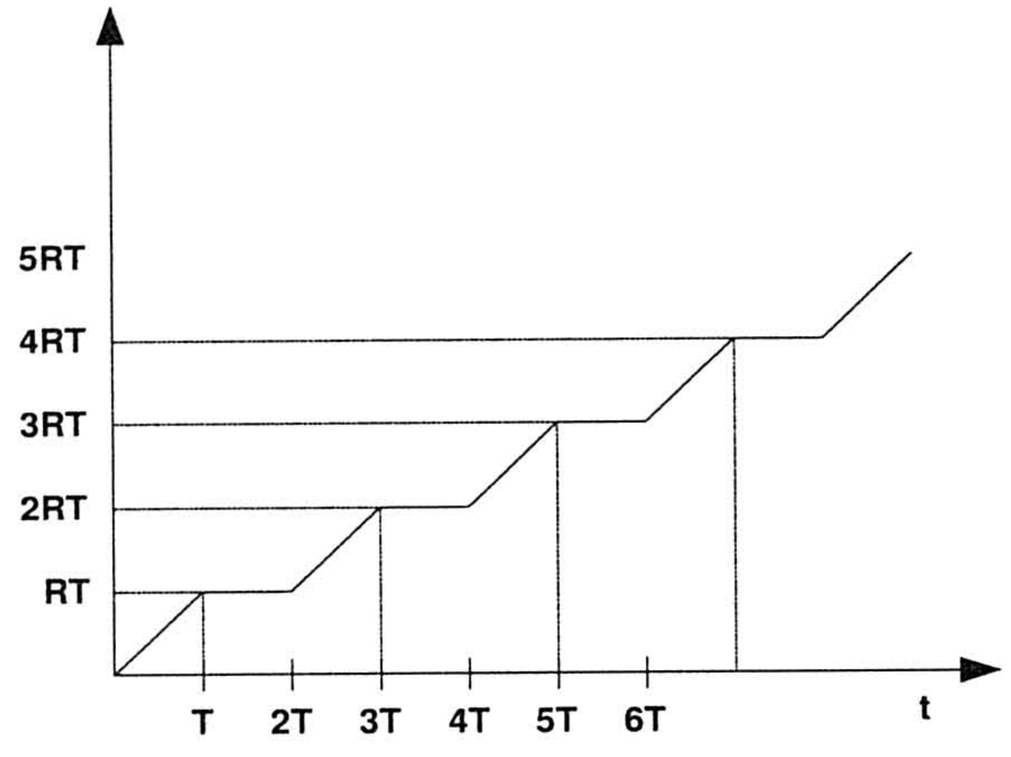

Figura A.3: Função composta.

A função base apresentada para o modelo é a representada na Figura A.3 que é similar às curvas das Figuras 5.4 (b) e 5.5 (b). Contudo, se está apenas uma fonte a transmitir então o modelo pode utilizar a função λ_R para representar a curva de saída; no entanto a existência de uma segunda fonte a iniciar a actividade em T obriga à distribuição da largura de banda pelas duas. Como a filosofia do modelo é conceder o recurso de forma a que a distribuição dos atrasos seja equitativa (de acordo com os objectivos definidos na secção 5.5), a segunda fonte deve ser servida durante o mesmo tempo da primeira e só depois é que a primeira fonte volta a ser servida e assim sucessivamente. Com base na Figura A.3 e fazendo $k = RT$ é possível verificar que a função que representa a curva é $ik + \beta_{R,2iT}$ para $i = 0, 1, 2, ...$, quando utilizadas as convoluções sucessivas e a respectiva obtenção de mínimos, isto é, a curva é de facto representada por

$$\inf_{i\geq 0}\{ik + \beta_{R,2iT}\} = \{\lambda_R\} \wedge \{k + \beta_{R,2T}\} \wedge \{2k + \beta_{R,4T}\} \wedge \{3k + \beta_{R,6T}\} \wedge ...$$

Se em vez de duas o número de fontes subir para três a função já apresenta uma forma diferente

$$\inf_{i\geq 0}\{ik+\beta_{R,3iT}\} = \{\lambda_R\} \wedge \{k+\beta_{R,3T}\} \wedge \{2k+\beta_{R,6T}\} \wedge \{3k+\beta_{R,9T}\} \wedge ...$$

e assim sucessivamente, com o caso geral dado por

$$\inf_{i>0}\{ik+\beta_{R,NiT}\} \tag{A.2}$$

em que N representa o número de fontes. Contudo, é necessário verificar que esta expressão apenas se verifica para situações em que as fonte iniciem a sua actividade em intervalos constantes de T, o que na realidade não é o que acontece na prática; no entanto fica justificado para um modelo em que se considerem apenas cenários deste tipo.

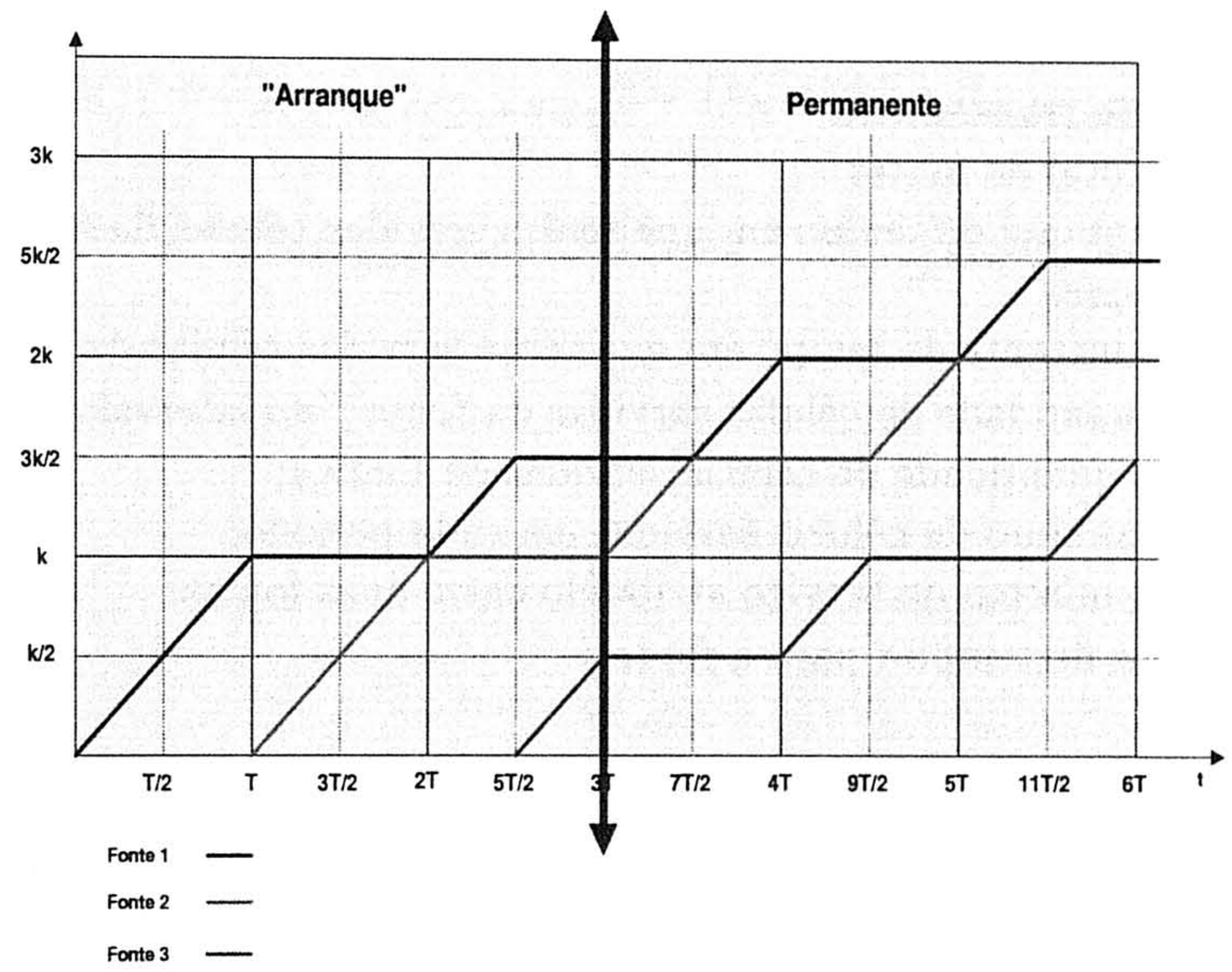

Figura A.4: Divisão das funções representativas das curvas em duas áreas distintas.

O problema coloca-se quando as fontes iniciam a sua actividade decaladas no tempo com intervalos diferentes. Antes de apresentar uma generalização da curva de saída é necessário explicar que a função obtida (função que representa a curva de saída) tem significado para os casos apresentados na secção 5.5 (fontes *On-Off* com períodos bem definidos) aquando da definição dos vários cenários. Nestas situações é possível verificar que é necessário definir dois momentos distintos (ver Figura A.4):

1. O primeiro, designado por "**Arranque**", acontece enquanto:

(a) não estiveram todas as fontes a transmitir;

(b) não estiver calculado o intervalo mínimo de transmissão entre fontes;

(c) não estiver calculado o número mínimo de células servidas para cada fonte em cada período;

2. O segundo momento desigando por "**Permanente**" ocorre assim que o sistema esteja estável (uma vez que já é conhecido o tamanho mínimo do intervalo de transmissão entre as fontes e o número exacto de células servidas para cada fonte).

Para a generalização, a função $f^j(t)$ representa o número de células servidas da fonte j com $j = 1, 2, ..., N$ (*curvas de saída*); no entanto, antes de apresentar a função é necessário definir o conjunto de variáveis utilizadas (ver Figura A.5).

j	número da fonte
i	período de transmissão
N	número total de fontes
LT^j	último instante de tempo em que foram servidas células da fonte j na *zona de arranque*;
IT^j	próximo instante de tempo em que serão servidas células da fonte j;
k^j_{i-1}	última quantidade de células servidas da fonte j no intervalo $i-1$;
k^j	primeira quantidade de células servidas da fonte j;
Δ^*_k	número mínimo de células servidas em cada período;
Δ_T	intervalo mínimo de serviço atribuído entre duas fontes;
T^j_i	intervalos de tempo i para a fonte j.

Figura A.5: Definição das variáveis para a definição da *curva de saída*.

A função que representa a *curva de saída* é definida pela expressão (A.3) para as N fontes com $j = 1, 2, ..., N$

$$f^j(t) = \begin{cases} ik^j_i + \beta_{R,iIT^j_i} & i = 0 \\ IT^j_i = LT^j_i \text{ e } k^j_i = k^j & i = 1 \\ k^j_{i-1} + \Delta^*_k + \beta_{R,IT^j_{i-1}+i\Delta_T} & 1 < i \leq N \\ k^j_{i-1} + \Delta^*_k + \beta_{R,IT^j_{i-1}+N\Delta_T} & i > N \end{cases} \quad \text{(A.3)}$$

Esta função desdobra-se para cada uma das curvas que representam o número de células servidas de cada um dos fluxos; deve ainda obedecer ao cálculo dos mínimos entre todas as parcelas obtidas até que seja possível desenhar as respectivas curvas, assim

$$f^j(t) = \inf_{i=0}\{ik_i^j+\beta_{R,iIT_i^j}\}\wedge \inf_{1<i\leq N}\{k_{i-1}^j+\Delta_k^*+\beta_{R,IT_{i-1}^j+i\Delta_T}\}\wedge \inf_{i>N}\{k_{i-1}^j+\Delta_k^*+\beta_{R,IT_{i-1}^j+N\Delta_T}\}$$

a primeira parcela é

$$\inf_{i=0}\{ik_i^j+\beta_{R,iIT_i^j}\} = \lambda_R^j$$

enquanto que as parcelas referentes à zona de **Arranque** são

$$\inf_{1<i\leq N}\{k_{i-1}^j+\Delta_k^*+\beta_{R,IT_{i-1}^j+i\Delta_T}\} = \{k_1^j+\Delta_k^*+\beta_{R,IT_1^j+2\Delta_T}\}\wedge\{k_2^j+\Delta_k^*+\beta_{R,IT_2^j+3\Delta_T}\}\wedge\ldots\wedge$$

$$\wedge\{k_{N-1}^j+\Delta_k^*+\beta_{R,IT_{N-1}^j+N\Delta_T}\}$$

e as referentes à zona **Permanente** são

$$\inf_{i>N}\{k_{i-1}^j+\Delta_k^*+\beta_{R,IT_{i-1}^j+N\Delta_T}\} = \{k_N^j+\Delta_k^*+\beta_{R,IT_N^j+N\Delta_T}\}\wedge$$

$$\wedge\{k_{N+1}^j+\Delta_k^*+\beta_{R,IT_{N+1}^j+N\Delta_T}\}\wedge\{k_{N+2}^j+\Delta_k^*+\beta_{R,IT_{N+2}^j+N\Delta_T}\}\wedge\ldots$$

Apêndice B

Componente *Edge Device*

A partir do momento em que se escolheu um simulador era necessário realizar as alterações necessárias de forma a conseguir simular o comportamento das propostas efectuadas com o objectivo de obter resultados para a sua validação.

Como observado ao longo da tese a maior intervenção foi realizada no componente ED, nomeadamente na implementação de algoritmos de escalonamento (os algoritmos apresentados no Capítulo 4 e no Capítulo 6) e na distribuição da largura de banda pelos VCs em competição (algoritmo apresentado no Capítulo 7).

Com esta variedade de intervenções era fácil de perceber que a interacção iria ser apenas com o componente Linhas Físicas (LF). Embora a intervenção se desse apenas com este tipo de componente o modo de funcionamento era igual entre qualquer componente existente no simulador, com excepção da interacção entre os BTEs e as aplicações. É, precisamente esse modo de funcionamento que era necessário perceber (que estruturas de dados eram utilizadas, que informação passava entre componentes, etc.) para que fosse possível construir um novo componente.

B.1 Estrutura base de um componente no simulador

Todos os componentes do simulador [50] têm uma *action routine* que é chamada por cada evento (usualmente não pelo gestor de eventos, mas pela *action routine* que escalonou o evento), para executar um conjunto de comandos que vão permitir ao componente ter um comportamento próprio. Os componentes podem enviar qualquer tipo de eventos (desde que sejam previamente definidos) para outro. Contudo, para permitir ao simulador

realizar várias funções de manutenção, as *action routines* devem responder a um conjunto mínimo de comandos. A estrutura base de um componente é a seguinte:

```
#include <sys/types.h>
#include <stdio.h>
#include <math.h>
#include "sim.h"
#include "mempool.h"
#include "simx.h"
#include "q.h"
#include "list.h"
#include "component.h"
#include "log.h"
#include "comptypes.h"
#include "cell.h"
#include "route.h"
#include "eventdefs.h"
#include "event.h"
#include "rbedgedevice.h"
#include "link.h"
#include "cbrconnection.h"
#include "vbrconnection.h"
#include "tcpconnection.h"
/* Conjunto de eventos para o proprio componente */
#define MY_RECEIVE (Evtype)(EV_CLASS_PRIVATE | 1)
#define MY_SLOT_TIME (Evtype)(EV_CLASS_PRIVATE | 2)
#define MY_COMPUT_TOKENS (Evtype)(EV_CLASS_PRIVATE | 8)
/* Definicao das rotinas especificas */
static caddr_t sw_reset(), sw_create(), sw_neighbor(), ...
/* A rotina principal do componente  */
caddr_t
edgedevice_action(src, sw, type, cell,vpi, arg)
     Component *src;
     register RBEdgeDevice *sw;
     int type;
     Cell *cell;
     VPI vpi;
     caddr_t arg;
{
  /* Tratamento dos eventos */
}
```

O local onde os eventos são tratados é referenciado na última linha comentada (/* Tratamento dos eventos */) no código anterior.

B.2 Estrutura base do componente ED

Com base na estrutura apresentada na secção anterior foi construído o componente ED. Antes propriamente de apresentar a listagem completa do código é necessário mostrar onde se realizou a intervenção. O código apresentado de seguida é extraído do componente implementado (o código completo pode ser observado na secção seguinte) em que se pode verificar os eventos que tiveram de ser definidos para simular o comportamento desejado.

```
switch (type)  {
  case EV_RESET:
    result = sw_reset(sw);
    break;
  case EV_CREATE:
    result = sw_create((char *)arg);
    break;
  case EV_DEL:
    comp_delete((Component *)sw);
    result = (caddr_t)sw;
    break;
  case EV_NEIGHBOR:
      result = sw_neighbor(sw, (Component *)arg);
      break;
    case EV_UNEIGHBOR:
      result = sw_uneighbor(sw, (Component *)arg);
      break;
    case EV_LEGAL_NEXT_HOPS:
      result = sw_hops(sw, (list *)arg);
      break;
    case EV_MAKE_ROUTE:
#ifdef DEBUG
dbg_write(debug_log, DBG_INFO, (Component *)sw,
  "processed EV_MAKE_ROUTE event (a no-op)");
#endif
      result = sw_route(sw, (list *) arg,vpi);
  break;
case EV_START:
  result = sw_tokenBucket_init(sw);
  ticks = USECS_TO_TICKS(424.0/(double)(pval(sw, sw_speed)->u.i));
  ev_enqueue(MY_SLOT_TIME, (Component *)sw, (Component *)sw,
         ev_now()+ticks, edgedevice_action,
 (Cell *)NULL,NULL, (caddr_t)NULL);
  ticks = USECS_TO_TICKS(1000);
  ev_enqueue(MY_RATE, (Component *)sw, (Component *)sw,
```

```
            ev_now()+ticks, edgedevice_action,
    (Cell *)NULL,NULL, (caddr_t)NULL);
     ticks = USECS_TO_TICKS(424.0/sw->IPOnOff);
     ev_enqueue(MY_CELL_SCHEDULING, (Component *)sw, (Component *)sw,
            ev_now()+ticks, edgedevice_action,
    (Cell *)NULL,NULL, (caddr_t)NULL);
     break;
  case MY_SLOT_TIME:
     ticks = USECS_TO_TICKS(424.0/(double)(pval(sw, sw_speed)->u.i));
     result = sw_proc_slot_time(sw);
     ev_enqueue(MY_SLOT_TIME, (Component *)sw, (Component *)sw,
        ev_now()+ticks, edgedevice_action,
  (Cell *)NULL,NULL, (caddr_t)NULL);
     break;
   case MY_COMPUT_TOKENS:
     result = sw_comput_tokens(sw,vpi);
     break;
   case MY_CELL_SCHEDULING:
     ticks = USECS_TO_TICKS(424.0/sw->IPOnOff);
     result = sw_cell_scheduling(sw);
     ev_enqueue(MY_CELL_SCHEDULING, (Component *)sw, (Component *)sw,
        ev_now()+ticks, edgedevice_action,
  (Cell *)NULL,NULL, (caddr_t)NULL);
     break;
  case MY_RATE:
     result = sw_comput_rate(sw);
     break;
  case EV_RECEIVE:
     result = sw_receive(sw, cell, src);
     break;
  case MY_RECEIVE:
     result = sw_my_receive(sw, cell, src);
     break;
  case EV_READY:
     result = sw_ready(sw, src);
     break;
  default:
     break;
  }  /* end switch */
```

O código apresentado serviu de base à construção do componente ED que tem uma estrutura arquitectónica que pode ser observada nas Figuras 5.2 e 5.3. O componente é constituído por um conjunto de filas de espera nas portas de entrada, um TB por cada porta de entrada e uma fila de espera a jusante de cada um dos TBs, uma fila de

espera relativamente à categoria de serviço onde os fluxos de tráfego IP são agregados e a fila de espera da porta de saída.

Além desta implementação, foram adicionadas disciplinas de serviço que servem as filas de espera a jusante dos TBs para a fila de espera da categoria de serviço. Foi ainda dada a possibilidade de armazenar todos os dados referentes aos vários parâmetros utilizados nas várias disciplinas de serviço para posterior tratamento estatístico. Além destas funcionalidades foi ainda adicionado ao componente um mecanismo de distribuição de largura de banda disponível pelas filas de espera que representam as várias categorias de serviço, uma vez que este componente vai servir para agregar fluxos IPmTB num VC ATM (funcionando como uma fonte e um destino virtual ATM), mas também como comutador ATM para VCs das várias categorias.

B.3 Listagem completa do componente *Edge Device*

```
/*************************************************
 *   rbedgedevice.c, v 3.0 r 1.0 2002/05/01    *
 *    Fernando Moreira        *
 *    Universidade Portucalense                *
 *************************************************/

#include <sys/types.h>
#include <stdio.h>
#include <math.h>
#include "sim.h"
#include "mempool.h"
#include "simx.h"
#include "q.h"
#include "list.h"
#include "component.h"
#include "log.h"
#include "comptypes.h"
#include "cell.h"
#include "route.h"
#include "eventdefs.h"
#include "event.h"
#include "rbedgedevice.h"
#include "link.h"
#include "cbrconnection.h"
#include "vbrconnection.h"
```

```
#include "tcpconnection.h"
extern tick_t warmuptime;
/* Events for myself */
#define MY_RECEIVE (Evtype)(EV_CLASS_PRIVATE | 1)
#define MY_SLOT_TIME (Evtype)(EV_CLASS_PRIVATE | 2)
#define TIMER (Evtype)(EV_CLASS_PRIVATE | 3)
#define MY_AI_TIME (Evtype)(EV_CLASS_PRIVATE | 7)
#define MY_COMPUT_TOKENS (Evtype)(EV_CLASS_PRIVATE | 8)
#define MY_COMPUT_PERIOD_ON (Evtype)(EV_CLASS_PRIVATE | 9)
#define MY_DONE_ON (Evtype)(EV_CLASS_PRIVATE | 10)
#define MY_CELL_SCHEDULING (Evtype)(EV_CLASS_PRIVATE | 11)
#define MY_RATE (Evtype)(EV_CLASS_PRIVATE | 12)
static caddr_t sw_reset(), sw_create(), sw_neighbor(), sw_uneighbor(),
  sw_hops(), sw_receive(), sw_my_receive(), sw_ready(),
  sw_schedule_output(), sw_rt_destroy(), sw_route(),sw_er(),
  sw_proc_slot_time(), sw_demux_spq(), sw_demux1_spq1(),
  sw_averaging_int_time(), sw_comput_tokens(),
  sw_schedule_off_period(), sw_comput_period_on(),
  sw_cell_scheduling(), sw_tokenBucket_init(),
  sw_comput_rate();
extern Log pkt_log;
#undef DEBUG
#ifdef DEBUG
extern Log debug_log;
#endif
/*OE*/
static tick_t ack_inter=155452;
static tick_t last_arrival;
static tick_t last_time;
/* The main action routine of a switch  */
/*ARGSUSED*/
caddr_t
edgedevice_action(src, sw, type, cell,vpi, arg)
     Component *src;
     register RBEdgeDevice *sw;
     int type;
     Cell *cell;
     VPI vpi;
     caddr_t arg;
{
  caddr_t result = NULL;
  int i,j,k;
  tick_t ticks, ticks1;
  dbg_set_level(DBG_INFO);
  /* Chust a big switch on type of event */
```

```
  switch (type)  {
    case EV_RESET:
      result = sw_reset(sw);
      break;
    case EV_CREATE:
      /* Minor sanity check -- sw should be NULL when initializing. */
#ifdef DEBUG
      if (sw)
dbg_write(debug_log, DBG_ERR, (Component *)NULL, "Switch initialization called with non-null pointer.");
#endif
      result = sw_create((char *)arg);
      break;
    case EV_DEL:
      comp_delete((Component *)sw);
      result = (caddr_t)sw; /* Just return something non-null */
      break;
    case EV_NEIGHBOR:
      result = sw_neighbor(sw, (Component *)arg);
      break;
    case EV_UNEIGHBOR:
      result = sw_uneighbor(sw, (Component *)arg);
      break;
    case EV_LEGAL_NEXT_HOPS:
      result = sw_hops(sw, (list *)arg);
      break;
    case EV_MAKE_ROUTE:
#ifdef DEBUG
dbg_write(debug_log, DBG_INFO, (Component *)sw, "processed EV_MAKE_ROUTE event (a no-op)");
#endif
      /* Have to return something non-NULL  */
      result = sw_route(sw, (list *) arg,vpi);
      break;
    case EV_START:
        result = sw_tokenBucket_init(sw);
       ticks = USECS_TO_TICKS(424.0/(double)(pval(sw, sw_speed)->u.i));
ev_enqueue(MY_SLOT_TIME, (Component *)sw, (Component *)sw,
            ev_now()+ticks, edgedevice_action, (Cell *)NULL,NULL, (caddr_t)NULL);
       ticks = USECS_TO_TICKS(1000);
ev_enqueue(MY_RATE, (Component *)sw, (Component *)sw,
            ev_now()+ticks, edgedevice_action, (Cell *)NULL,NULL, (caddr_t)NULL);
       ticks = USECS_TO_TICKS(424.0/sw->IPOnOff);
 ev_enqueue(MY_CELL_SCHEDULING, (Component *)sw, (Component *)sw,
            ev_now()+ticks, edgedevice_action, (Cell *)NULL,NULL, (caddr_t)NULL);
      if (sw->sw_operation->u.i == 7){
         sw->period_on_initial = 1;
```

```
        ticks = USECS_TO_TICKS(1000);
        ev_enqueue(MY_COMPUT_PERIOD_ON, (Component *)sw, (Component *)sw,
            ev_now()+ticks, edgedevice_action, (Cell *)NULL,NULL, (caddr_t)NULL);
    }
    break;
  case MY_SLOT_TIME:
    ticks = USECS_TO_TICKS(424.0/(double)(pval(sw, sw_speed)->u.i));
    result = sw_proc_slot_time(sw);
    ev_enqueue(MY_SLOT_TIME, (Component *)sw, (Component *)sw,
       ev_now()+ticks, edgedevice_action, (Cell *)NULL,NULL, (caddr_t)NULL);
    break;
  case MY_AI_TIME:
    ticks =  USECS_TO_TICKS(pval(sw, sw_AI)->u.d);
    result = sw_averaging_int_time(sw);
    ev_enqueue(MY_AI_TIME, (Component *)sw, (Component *)sw,
       ev_now()+ticks, edgedevice_action, (Cell *)NULL,NULL, (caddr_t)NULL);
    break;
  case MY_COMPUT_TOKENS:
    result = sw_comput_tokens(sw,vpi);
    break;
   case MY_COMPUT_PERIOD_ON:
    result = sw_comput_period_on(sw,vpi);
    break;
  case MY_DONE_ON:
    result = sw_schedule_off_period(sw,vpi);
    break;
  case MY_CELL_SCHEDULING:
     ticks = USECS_TO_TICKS(424.0/sw->IPOnOff);
    result = sw_cell_scheduling(sw);
    ev_enqueue(MY_CELL_SCHEDULING, (Component *)sw, (Component *)sw,
       ev_now()+ticks, edgedevice_action, (Cell *)NULL,NULL, (caddr_t)NULL);
    break;
  case MY_RATE:
    result = sw_comput_rate(sw);
    break;
  case EV_RECEIVE:
    result = sw_receive(sw, cell, src);
    break;
  case MY_RECEIVE:
    result = sw_my_receive(sw, cell, src);
    break;
  case EV_READY:
    result = sw_ready(sw, src);
    break;
  default:
```

```
#ifdef DEBUG
      fprintf(stdout, "edgedevice_action: received unrecognized event of type %d\n", type);
      fflush(stdout);
#endif
      break;
    }  /* end switch */
  return(result);
}
/*****************************************/
static caddr_t
sw_reset(sw)
     register RBEdgeDevice *sw;
{
  register Neighbor *n;
  int i,j,k;
  int num_setup_ABR;
  double fair_share;
  for (n = (Neighbor *)sw->sw_neighbors->l_head; n; n = n->n_next){
     n->n_busy = FALSE;
  }
  /* Clear all shared memory cell queues  */
  for (i=0;i<MAXPORT;i++){
    if (sw->voport[i].to_link){
     if(sw->voport[i].spq) while (q_deq(sw->voport[i].spq));
     if(sw->voport[i].spq1) while (q_deq(sw->voport[i].spq1));
       sw->voport[i].link_busy = FALSE;
       sw->voport[i].vbr.congested = FALSE;
       sw->voport[i].abr.congested = FALSE;
       sw->voport[i].abrIP.congested = FALSE;
       sw->voport[i].abrIP.state= IDLE;
       sw->voport[i].abrOut.congested = FALSE;
       sw->voport[i].abrOut.state= IDLE;
       sw->voport[i].abr.very_congested = FALSE;
sw->voport[i].vbr.iq->u.d = 0.0;
       log_param((Component *)sw, pval(sw, voport[i].vbr.iq));
       sw->voport[i].vbr.iq->u.sample = ev_now();
       sw->voport[i].vbr.dq->u.d = 0.0;
log_param((Component *)sw, pval(sw, voport[i].vbr.dq));
sw->voport[i].vbr.dq->u.sample = ev_now();
while (q_deq(sw->voport[i].vbr.ptr)) ;
sw->voport[i].abr.iq->u.i = 0;
log_param((Component *)sw, pval(sw, voport[i].abr.iq));
sw->voport[i].abr.iq->u.sample = ev_now();
sw->voport[i].abr.dq->u.d = 0.0;
log_param((Component *)sw, pval(sw, voport[i].abr.dq));
```

```
sw->voport[i].abr.dq->u.sample = ev_now();
while (q_deq(sw->voport[i].abr.ptr)) ;
sw->voport[i].abrIP.iq->u.i = 0;
log_param((Component *)sw, pval(sw, voport[i].abrIP.iq));
sw->voport[i].abrIP.iq->u.sample = ev_now();
sw->voport[i].abrIP.dq->u.d = 0.0;
log_param((Component *)sw, pval(sw, voport[i].abrIP.dq));
sw->voport[i].abrIP.dq->u.sample = ev_now();
while (q_deq(sw->voport[i].abrIP.ptr)) ;
sw->voport[i].abrOut.iq->u.i = 0;
log_param((Component *)sw, pval(sw, voport[i].abrOut.iq));
sw->voport[i].abrOut.iq->u.sample = ev_now();
sw->voport[i].abrOut.dq->u.i = 0;
log_param((Component *)sw, pval(sw, voport[i].abrOut.dq));
sw->voport[i].abrOut.dq->u.sample = ev_now();
while (q_deq(sw->voport[i].abrOut.ptr)) ;
sw->voport[i].abr.count = 0;
sw->voport[i].abr.count1 = 0;
sw->voport[i].abr.count3 = 0;
sw->voport[i].abr.rate = sw->sw_TR;
sw->voport[i].abr.stamp = ev_now();
sw->voport[i].abr.q_level = 0;
sw->voport[i].abr.tokens = 0;
sw->voport[i].abr.b = 0;
sw->voport[i].abr.x = 0;
sw->voport[i].abr.dly = 0;
sw->voport[i].abr.delay = 0.0;
sw->voport[i].abr.xOld = 0;
sw->voport[i].abr.flag = 0;
sw->voport[i].abr.flagA = 0;
sw->voport[i].abr.celulasPorta = 0;
sw->voport[i].abr.vci = 0.0;
sw->voport[i].abr.vpi = -1;
sw->voport[i].abr.on = 1;
sw->voport[i].abr.controlo = 0;
sw->voport[i].abr.burstInitTime = 0.0;
sw->voport[i].abr.burstInit = 0;
sw->voport[i].abr.burstTransmissionTime = 0.0;
sw->voport[i].abr.intervalMin = 0.0;
sw->voport[i].abr.credits = 0;
sw->voport[i].abr.creditsAcum = 0;
sw->voport[i].abr.ERS->u.d = sw->sw_TR->u.d;
log_param((Component *)sw, pval(sw, voport[i].abr.ERS));
sw->voport[i].abr.ERS->u.sample = ev_now();
sw->voport[i].abr.Rate_Abr->u.d = 1.0;
```

```
log_param((Component *)sw, pval(sw, voport[i].abr.Rate_Abr));
sw->voport[i].abr.k->u.d = 0.0;
log_param((Component *)sw, pval(sw, voport[i].abr.k));
sw->voport[i].abr.count_abr->u.d=0.0;
log_param((Component *)sw, pval(sw, voport[i].abr.count_abr));
sw->voport[i].abr.q_old = 0;
sw->voport[i].abr.d = 0;
sw->voport[i].abr.celulas_abr = 0;
      for (j=0;j<MXVC;j++){
       sw->voport[i].abr.FerRuela[j].TotalCelulas = 0;
       sw->voport[i].abr.FerRuela[j].ERS = 0.0;
       sw->voport[i].abr.FerRuela[j].portaI = -1;
       sw->voport[i].abr.FerRuela[j].portaO = -1;
       sw->voport[i].abr.FerRuela[j].portaControlo = -1;
      }
  sw->voport[i].vbr.celulas_vbr = 0;
        sw->voport[i].vbr.count_vbr->u.d=0.0;
        log_param((Component *)sw, pval(sw, voport[i].vbr.count_vbr));
        sw->voport[i].vbr.Rate_Vbr->u.d = 0.0;
        log_param((Component *)sw, pval(sw, voport[i].vbr.Rate_Vbr));
        sw->voport[i].vbr.rate_vbr = 0.0;
/* numero inicial de tokens */
      for (j=0;j<MXVC;j++){
if(sw->voport[i].abr.FerRuela[j].vpi != -1)
sw->voport[i].abr.tokens=5;
      }
        sw->voport[i].abr.q_old = 0;
        sw->voport[i].abr.d = 0;
        sw->voport[i].abr.celulas_abr = 0;
    } /* if connected */
    sw->Zigual[i] = 0;
    sw->y[i] = 0;
  } /* i loop */
  pval(sw, sw_dropped)->u.d = 0.0;
  sw->num_dropped=0;
  sw->total_cell=0;
  log_param((Component *)sw, pval(sw, sw_dropped));
  sw->sw_dropped->u.sample = ev_now();
  sw->outputPort=-1;
  sw->period_on_initial = 1;
  sw->IPOnOff = sw->sw_TR->u.d;
  sw->IPOnOff1 = sw->sw_TR->u.d;
  sw->flag=0;
  sw->timeInterval = 0.0;
  sw->cellSchedulingTime = 424.0/sw->IPOnOff;
```

```
  sw->tFlag=0;
  sw->flagX=0;
  sw->dtime=0.0;
  sw->intCredits=0;
  sw->creditsMin=10000;
  sw->creditsMin1=sw->creditsMin;
  sw->iq = sw->jv = 0;
#ifdef DEBUG
  fprintf(stdout,"sw_reset %g\n",ev_now());
  fflush(stdout);
#endif
  return((caddr_t)sw);
}
/****************************************/
static caddr_t
sw_create(name)
     register char *name;
{
  RBEdgeDevice *newsw;
  int i,j,k;
  newsw = (RBEdgeDevice *)malloc(sizeof(RBEdgeDevice));
  newsw->sw_neighbors = l_create();
  strncpy(newsw->sw_name, name, 40);
  newsw->sw_params = q_create();
  newsw->sw_type = EDGEDEVICE;
  newsw->sw_class = SWITCH_CLASS;
  newsw->sw_action = edgedevice_action;
  newsw->sw_menu_up = FALSE;
  for (i=0;i<MAXPORT;i++){
    newsw->voport[i].to_link = (Component *)NULL;
    newsw->voport[i].vbr.ptr= (queue *)NULL;
    newsw->voport[i].abr.ptr= (queue *)NULL;
    newsw->voport[i].abrIP.ptr= (queue *)NULL;
    newsw->voport[i].abrIP.state= IDLE;
    newsw->voport[i].abrOut.ptr= (queue *)NULL;
    newsw->voport[i].abrOut.state= IDLE;
    newsw->voport[i].vbr.congested = FALSE;
    newsw->voport[i].abr.congested = FALSE;
    newsw->voport[i].abr.very_congested = FALSE;
    newsw->voport[i].abr.count = 0;
    newsw->voport[i].abr.count1 = 0;
    newsw->voport[i].abr.count3 = 0;
    newsw->voport[i].abr.celulas_abr = 0;
    newsw->voport[i].abr.q_level = 0;
    newsw->voport[i].abr.q_old = 0;
```

```
    newsw->voport[i].abr.d = 0;
    newsw->voport[i].abr.SCRd = 0.0;
    newsw->voport[i].abr.vci = 0.0;
    newsw->voport[i].abr.vpi = -1;
    newsw->voport[i].abr.on = 1;
    newsw->voport[i].abr.x = 0;
    newsw->voport[i].abr.dly = 0;
    newsw->voport[i].abr.delay = 0.0;
    newsw->voport[i].abr.xOld = 0;
    newsw->voport[i].abr.flag = 0;
    newsw->voport[i].abr.flagA = 0;
    newsw->voport[i].abr.b = 0;
    newsw->voport[i].abr.controlo = 0;
    newsw->voport[i].abr.credits = 0;
    newsw->voport[i].abr.creditsAcum = 0;
    newsw->Zigual[i] = 0;
    newsw->y[i] = 0;
    newsw->voport[i].abr.burstInitTime = 0.0;
    newsw->voport[i].abr.burstTransmissionTime = 0.0;
    newsw->voport[i].abr.burstInit = 0;
    newsw->voport[i].abr.intervalMin = 0.0;

  } /* i */

  newsw->sw_cellog = param_init((Component *)newsw, "Name",
     (PFD)NULL, make_name_text, make_short_name_text,
     param_input_name,
     0, DisplayMask | InputMask | CanHaveLogMask, 0.0);
  newsw->sw_delay = param_init((Component *)newsw,
    "Delay to process a cell (uSec)",
     (PFD)NULL, make_int_text, make_short_int_text,
     param_input_int,
     0, DisplayMask | InputMask, 0.0);
  newsw->sw_speed =  param_init((Component *)newsw,
           "Slot time (Mbit/s)",
            (PFD)NULL, make_int_text, make_short_int_text,
            param_input_int,
            0, DisplayMask | InputMask, 0.0);
  pval(newsw, sw_speed)->u.i = 155;
  newsw->sw_oqsize = param_init((Component *)newsw,
    "Output q_size (cells, -1=inf)",
     (PFD)NULL, make_int_text, make_short_int_text,
     param_input_int,
     0, DisplayMask | InputMask, 0.0);
  pval(newsw, sw_oqsize)->u.i = -1;
```

```
newsw->sw_ht = param_init((Component *)newsw,
          "High Threshold (Cells)",
           (PFD)NULL, make_int_text, make_short_int_text,
           param_input_int,
           0, DisplayMask | InputMask, 0.0);
pval(newsw, sw_ht)->u.i = 80;
newsw->sw_lt = param_init((Component *)newsw,
          "Low Threshold (Cells)",
           (PFD)NULL, make_int_text, make_short_int_text,
           param_input_int,
           0, DisplayMask | InputMask, 0.0);
  pval(newsw, sw_lt)->u.i = 60;
  newsw->sw_log_factor = param_init((Component *)newsw,
           "Logging every (ticks) (e.g. 1, 100)",
           (PFD)NULL,make_int_text, make_short_int_text,
           param_input_int,
           0, DisplayMask | InputMask, 0.0);
pval(newsw, sw_log_factor)->u.i = 1000;
   newsw->sw_operation = param_init((Component *)newsw,
           "7 (FerRuela)",
           (PFD)NULL,make_int_text, make_short_int_text,
           param_input_int,
           0, DisplayMask | InputMask, 0.0);
 pval(newsw, sw_operation)->u.i = 7;
  newsw->sw_TR = param_init((Component *)newsw,
           "Target Rate (Mbits/s)",
           (PFD)NULL,make_double_text, make_short_double_text,
           param_input_double,
           0, DisplayMask | InputMask, 0.0);
  pval(newsw, sw_TR)->u.d = 147;
  newsw->sw_N = param_init((Component *)newsw,
           "Measurement Interval in cells (N)",
           (PFD)NULL,make_int_text, make_short_int_text,
           param_input_int,
           0, DisplayMask | InputMask, 0.0);
  pval(newsw, sw_N)->u.i = 100;
   newsw->sw_tau = param_init((Component *)newsw,
           "Congestion Tolerance in cells (tau)",
           (PFD)NULL,make_int_text, make_short_int_text,
           param_input_int,
           0, DisplayMask | InputMask, 0.0);
pval(newsw, sw_tau)->u.i = 0;
newsw->sw_congestion = param_init((Component *)newsw,
          "Congestion (True/False)",
           int_calc, make_int_text, make_short_int_text,
```

```
            (PFI)NULL, BINARY,
            DisplayMask | CanHaveMeterMask | CanHaveLogMask, 1.0);
pval(newsw, sw_congestion)->u.i = 0;
newsw->sw_dropped = param_init((Component *)newsw,
  "% cell drop",
   double_calc, make_double_text, make_short_double_text,
   (PFI)NULL, TIME_HISTORY,
   DisplayMask | CanHaveMeterMask | CanHaveLogMask, 1.0);
pval(newsw, sw_dropped)->u.d = 0.0;
newsw->num_dropped=0;
newsw->total_cell=0;
newsw->sw_BRM_rate = param_init((Component *)newsw,
          "Max BRM rate (cells/sec)",
           (PFD)NULL, make_double_text, make_short_double_text,
           param_input_double, 0,
           DisplayMask | InputMask, 0.0);

newsw->routing_table=ht_create(1117,sizeof(Route_key),sizeof(Route_rec));
newsw->sw_AI = param_init((Component *)newsw,
   "Averaging Interval in usecs (AI)",
           (PFD)NULL,make_double_text, make_short_double_text,
           param_input_double,
            0, DisplayMask | InputMask, 0.0);
pval(newsw, sw_AI)->u.d = 5000;
newsw->sw_countvbr = param_init((Component *)newsw,
         "Fernando - Numero de celulas VBR",
         int_calc, make_int_text, make_short_int_text,
         (PFI)NULL, TIME_HISTORY,
         DisplayMask | CanHaveMeterMask | CanHaveLogMask, 1.0);
    pval(newsw, sw_countvbr)->u.i = 0;
      newsw->sw_ratevbr = param_init((Component *)newsw,
                "Fernando - Rate do VBR",
                double_calc, make_double_text, make_short_double_text,
                (PFI)NULL, TIME_HISTORY,
                DisplayMask | CanHaveMeterMask | CanHaveLogMask, 1.0);
    pval(newsw, sw_ratevbr)->u.d = 0.0;
  newsw->TRaux=0.0;
  newsw->tFlag=0;
  newsw->flagX=0;
  newsw->dtime=0.0;
  newsw->flag=0;
  newsw->E1=0.0;
  newsw->E2=0.0;
  newsw->E21=0.0;
  newsw->E22=0.0;
```

```
    newsw->E3=0.0;
    newsw->E4=0.0;
    newsw->E10=0.0;
    newsw->E11=0.0;
    newsw->acr=0.0;
    newsw->AR=0.0;
    newsw->TotalMCR=1.0;
    newsw->TotalPCR=1.0;
    newsw->outputPort=-1;
    newsw->period_on_initial = 0;
    newsw->IPOnOff = newsw->sw_TR->u.d;
    newsw->IPOnOff1 = newsw->sw_TR->u.d;
    newsw->interval = 50500.0;
    newsw->interval1 = newsw->interval;
    newsw->timeInterval = 0.0;
    newsw->cellSchedulingTime = 424.0/newsw->IPOnOff;
    newsw->intCredits = 0;
    newsw->creditsMin = 10000;
    newsw->creditsMin1 = newsw->creditsMin;
#ifdef DEBUG
  fprintf(stdout,"%s switch initialized \n",newsw->sw_name);
  fflush(stdout);
#endif
  return((caddr_t)newsw);
}

/*****************************************/
static caddr_t
sw_route(sw,route_list,vpi)
        register RBEdgeDevice *sw;
         list *route_list;
         VPI vpi;
{

Route_key rk;
Route_rec rr;
l_elt *le;
int result;
int oneway;
Component *which_link,*which_link2;
Component *c1;
register Neighbor *n;
Param *p;
char local_vpi[5];
int connection_type;
```

```
int i, j;
if (((Component *) route_list->l_head->le_data)->co_class!=CONNECTION ||
    ((Component *) route_list->l_tail->le_data)->co_class!=CONNECTION) {
#ifdef DEBUG
    dbg_write(debug_log, DBG_INFO, (Component *)sw,
              "Error: one or both ends of the route are not CONNECTIONS");
#endif
    return(FALSE);
}
connection_type = ((Component *) route_list->l_head->le_data)->co_type ;
rr.rr_route=route_list;
rk.rk_vpi = vpi;
for (le=route_list->l_head;le;le=le->le_next) {
    if (sw==((RBEdgeDevice *) le->le_data)) {
       rk.rk_last_link = (Component *)le->le_prev->le_data;
       rr.rr_info.next_hop=rr.rr_info.next_link=NULL;
       if (le->le_next) {
          rr.rr_info.next_link=(Component *) le->le_next->le_data;
          which_link = (Component *)le->le_next->le_data;
          /* OE modification */
            if (connection_type==TCPCONNECTION)
                    which_link->number_of_vcs=vpi+1;
          if (le->le_next->le_next)
              rr.rr_info.next_hop=(Component *) le->le_next->le_next->le_data;
              rr.rr_info.connection_type = connection_type;
              rr.rr_info.VC_CI = 0;
      rr.rr_info.number_of_cells = 0;
            }
         result=ht_insert(sw->routing_table,(char*) &rk,(char*) &rr);
         if (result==-1) {
#ifdef DEBUG
dbg_write(debug_log,DBG_INFO,(Component*)NULL,"sw_route: ht_insert failed
            -- routing loop");
#endif
} } }
rk.rk_last_link = NULL;
    for (le=route_list->l_tail;le;le=le->le_prev) {
  if (sw==(RBEdgeDevice *) le->le_data) {
    rk.rk_last_link = (Component *)le->le_next->le_data;
     rr.rr_info.next_hop=rr.rr_info.next_link=NULL;
     if (le->le_prev) {
        which_link2 = (Component *)le->le_prev->le_data;
        rr.rr_info.next_link =(Component *)le->le_prev->le_data;
        if (le->le_prev->le_prev)
          rr.rr_info.next_hop=(Component *)le->le_prev->le_prev->le_data;
```

```
                rr.rr_info.connection_type = connection_type;
                rr.rr_info.VC_CI = 0;
 rr.rr_info.number_of_cells = 0;
              }
            result=ht_insert(sw->routing_table,(char *)&rk,(char *)&rr);
            if (result==-1) {
#ifdef DEBUG
       dbg_write(debug_log,DBG_INFO,(Component *) NULL,
                 "sw_route: ht_insert failed --routing loop 2");
#endif
} } }
/* And that´s it ! */

    if ((connection_type == ABRCONNECTION1) || (connection_type == ABRCONNECTION4)) {
      /* Direction forward */
      /* locate the output port */
      for (le=route_list->l_head;le;le=le->le_next)
if (sw==((RBEdgeDevice *) le->le_data)) {
  for (i=0;i<MAXPORT;i++)
    if (sw->voport[i].to_link == (Component *) le->le_next->le_data) break;
  break;
      }
/* find an empty FerRuela element */
      for (j=0;j<MXVC;j++){
if(sw->voport[i].abr.FerRuela[j].vpi == -1) break;
      }
      sw->voport[i].abr.FerRuela[vpi].vpi =vpi;
      /* Direction backward */
      /* locate the output port */
      for (le=route_list->l_tail;le;le=le->le_prev)
if (sw==((RBEdgeDevice *) le->le_data)) {
  for (i=0;i<MAXPORT;i++)
    if (sw->voport[i].to_link == (Component *) le->le_prev->le_data) break;
  break;
}
/* find an empty FerRuela element */
      for (j=0;j<MXVC;j++){
if(sw->voport[i].abr.FerRuela[j].vpi == -1) break;
      }
      sw->voport[i].abr.FerRuela[vpi].vpi =vpi;
    }
return((caddr_t)sw);
}
/*****************************************/
```

```
static caddr_t
sw_neighbor(sw, c)
     register RBEdgeDevice *sw;
     register Component *c;
{
  register Neighbor *n;
  Param *p;
  int i,j;
  if (!(n = add_neighbor(sw, c, 0, 1, LINK)))
    return((caddr_t)NULL);
  n->n_busy = FALSE;
  for (i=0;i<MAXPORT;i++){ /* find an empty spot */
    if(!sw->voport[i].to_link) break;
  }
  if(i >= MAXPORT){ /* no more output queue spot */
#ifdef DEBUG
    fprintf(stdout,"sw_neighbor: No more output queue, too many neighbors\n");
    fflush(stdout);
    return((caddr_t)NULL);
#endif
  } else {
    sw->voport[i].link_busy = FALSE;
    sw->voport[i].to_link = c;
sw->voport[i].spq = q_create();
sw->voport[i].spq1 = q_create();
 sw->voport[i].vbr.ptr = q_create();
 sw->voport[i].abr.ptr = q_create();
 sw->voport[i].abrIP.ptr = q_create();
 sw->voport[i].abrOut.ptr = q_create();
 sw->voport[i].vbr.iq =
            param_init((Component *)sw, "Delay in Q to ",
             double_calc, make_double_text, make_short_double_text,
            (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
               | CanHaveLogMask, 0.0);
 strcat(sw->voport[i].vbr.iq->p_name, sw->voport[i].to_link->co_name);
 sw->voport[i].vbr.dq =
            param_init((Component *)sw, "Input rate for AbrIP from ",
             double_calc, make_double_text, make_short_double_text,
            (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
               | CanHaveLogMask, 0.0);
  strcat(sw->voport[i].vbr.dq->p_name,  sw->voport[i].to_link->co_name);
sw->voport[i].abr.iq =
            param_init((Component *)sw, "Cells in ABR Q to ",
             int_calc, make_int_text, make_short_int_text,
            (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
```

```
                | CanHaveLogMask, 0.0);
        strcat(sw->voport[i].abr.iq->p_name,
                sw->voport[i].to_link->co_name);
        sw->voport[i].abr.dq =
                param_init((Component *)sw, "X value to ",
                 double_calc, make_double_text, make_short_double_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                   | CanHaveLogMask, 0.0);
         strcat(sw->voport[i].abr.dq->p_name,
                 sw->voport[i].to_link->co_name);
        sw->voport[i].abrIP.iq =
                param_init((Component *)sw, "Cells in ABR(I) Q TB to ",
                 int_calc, make_int_text, make_short_int_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                   | CanHaveLogMask, 0.0);
         strcat(sw->voport[i].abrIP.iq->p_name,
                 sw->voport[i].to_link->co_name);
         sw->voport[i].abrIP.dq =
                param_init((Component *)sw, "Cell delay to ",
                 double_calc, make_double_text, make_short_double_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                   | CanHaveLogMask, 0.0);
          strcat(sw->voport[i].abrIP.dq->p_name, sw->voport[i].to_link->co_name);
        sw->voport[i].abrOut.iq =
                param_init((Component *)sw, "Cells in ABR(I) Q MTB to ",
                 int_calc, make_int_text, make_short_int_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                   | CanHaveLogMask, 0.0);
         strcat(sw->voport[i].abrOut.iq->p_name,
                 sw->voport[i].to_link->co_name);
         sw->voport[i].abrOut.dq =
                param_init((Component *)sw, "Tokens available (I) to ",
                 int_calc, make_int_text, make_short_int_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                   | CanHaveLogMask, 0.0);
          strcat(sw->voport[i].abrOut.dq->p_name,
                 sw->voport[i].to_link->co_name);
           sw->voport[i].abr.ERS =
                param_init((Component *)sw, "ERS for Q to",
                 double_calc, make_double_text, make_short_double_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                   | CanHaveLogMask, 0.0);
              strcat(sw->voport[i].abr.ERS->p_name, sw->voport[i].to_link->co_name);
        sw->voport[i].abr.ERS->u.d = sw->sw_TR->u.d;
           sw->voport[i].abr.Rate_Abr =
```

```
                    param_init((Component *)sw, "Rate AbrIP for Q to ",
                    double_calc, make_double_text, make_short_double_text,
                    (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                   | CanHaveLogMask, 0.0);
     strcat(sw->voport[i].abr.Rate_Abr->p_name, sw->voport[i].to_link->co_name);
sw->voport[i].vbr.Rate_Vbr =
       param_init((Component *)sw, "Rate VBR for Q to ",
                    double_calc, make_double_text, make_short_double_text,
                    (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                    | CanHaveLogMask, 0.0);
     strcat(sw->voport[i].vbr.Rate_Vbr->p_name,
sw->voport[i].to_link->co_name);
        sw->voport[i].abr.k =
                     param_init((Component *)sw, "Oldest Cell for Q to ",
                     double_calc, make_double_text, make_short_double_text,
                     (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                     | CanHaveLogMask, 0.0);
         strcat(sw->voport[i].abr.k->p_name, sw->voport[i].to_link->co_name);
         strcat(sw->voport[i].abr.Fr->p_name, sw->voport[i].to_link->co_name);
        sw->voport[i].abr.w1 =
                     param_init((Component *)sw, "Accumulated Z to ",
                     int_calc, make_int_text, make_short_int_text,
                     (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                     | CanHaveLogMask, 0.0);

 strcat(sw->voport[i].abr.w1->p_name, sw->voport[i].to_link->co_name);
sw->voport[i].abr.w2 =
                param_init((Component *)sw, "Accumulated X to ",
                int_calc, make_int_text, make_short_int_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                | CanHaveLogMask, 0.0);
 strcat(sw->voport[i].abr.w2->p_name, sw->voport[i].to_link->co_name);
sw->voport[i].abr.Fr =
                param_init((Component *)sw, "Accumulated Q to ",
                double_calc, make_double_text, make_short_double_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                | CanHaveLogMask, 0.0);
        strcat(sw->voport[i].abr.Fr->p_name, sw->voport[i].to_link->co_name);
   sw->voport[i].abr.count_abr =
                 param_init((Component *)sw, "count ABR cell for Q to ",
                 double_calc, make_double_text, make_short_double_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                 | CanHaveLogMask, 0.0);
  strcat(sw->voport[i].abr.count_abr->p_name, sw->voport[i].to_link->co_name);
 sw->voport[i].vbr.count_vbr =
```

```
                param_init((Component *)sw, "count VBR cell for Q to ",
                double_calc, make_double_text, make_short_double_text,
                (PFI)NULL, TIME_HISTORY, DisplayMask | CanHaveMeterMask
                | CanHaveLogMask, 0.0);
strcat(sw->voport[i].vbr.count_vbr->p_name, sw->voport[i].to_link->co_name);
 sw->voport[i].abr.SCR =
                param_init((Component *)sw, "Sustained Cell Rate (SCR) to ",
                double_calc, make_double_text, make_short_double_text,
                param_input_double, 0, DisplayMask | InputMask, 0.0);
   strcat(sw->voport[i].abr.SCR->p_name, sw->voport[i].to_link->co_name);
   sw->voport[i].abr.PCR =
                param_init((Component *)sw, "Peak Cell Rate (PCR) to ",
                double_calc, make_double_text, make_short_double_text,
                param_input_double, 0, DisplayMask | InputMask, 0.0);
   strcat(sw->voport[i].abr.PCR->p_name, sw->voport[i].to_link->co_name);
   sw->voport[i].abr.MBS =
                param_init((Component *)sw, "Maximum Burst Size (MBS) to ",
                int_calc, make_int_text, make_short_int_text,
                param_input_int, 0, DisplayMask | InputMask, 0.0);
   strcat(sw->voport[i].abr.MBS->p_name, sw->voport[i].to_link->co_name);
     sw->period_on_initial++;
/* Initialization of FerRuela info */
        for (j=0; j<MXVC;j++){
            sw->voport[i].abr.FerRuela[j].vpi = -1;
            sw->voport[i].abr.FerRuela[j].portaI = -1;
            sw->voport[i].abr.FerRuela[j].portaO = -1;
            sw->voport[i].abr.FerRuela[j].portaControlo = -1;
            sw->voport[i].abr.FerRuela[j].ERS = 0.0;
}
#ifdef DEBUG
    fprintf(stdout,"sw_neighbor:sw=%d, voport[%d]=%d initialized\n",
        sw,i,sw->voport[i].to_link);
    fflush(stdout);
#endif
  }
  return((caddr_t)n);
}
/****************************************/
static caddr_t
sw_uneighbor(sw, c)
     register RBEdgeDevice *sw;
     register Component *c;
{
  int i,j,k;
  for (i=0;i<MAXPORT;i++){
```

```
if(sw->voport[i].to_link == c){
  sw->voport[i].to_link = (Component *)NULL;
  if (sw->voport[i].spq){
    while (q_deq(sw->voport[i].spq)) ;
    while (q_deq(sw->voport[i].spq1)) ;
    while (q_deq(sw->voport[i].vbr.ptr));
    while (q_deq(sw->voport[i].abr.ptr));
    while (q_deq(sw->voport[i].abrIP.ptr));
    while (q_deq(sw->voport[i].abrOut.ptr));
      free(sw->voport[i].spq);
      free(sw->voport[i].spq1);
      free(sw->voport[i].vbr.ptr);
      free(sw->voport[i].abr.ptr);
      free(sw->voport[i].abrIP.ptr);
      free(sw->voport[i].abrOut.ptr);
    }
  }
  }
  return((caddr_t)remove_neighbor((Component *)sw, c));
}
/*****************************************/
static caddr_t
sw_hops(sw, l)
    register RBEdgeDevice *sw;
    register list *l;
{
  register Component *c;
  register list *nlist;
  register Neighbor *n;
  c = (Component *)(l->l_tail);
  nlist = l_create();
  for (n = (Neighbor *)sw->sw_neighbors->l_head; n; n = n->n_next)
    if (n->n_c != c)
      l_addt(nlist, (caddr_t)n->n_c);
  return((caddr_t)nlist);
}
/*****************************************/
static caddr_t
sw_receive(sw, cell, src)
    register RBEdgeDevice *sw;
    register Cell *cell;
    register Component *src;
{
  register Neighbor *n;
  register int ticks;
```

```
register VPI vpi;
register Component *c;
register char src_name[40];
register int i;
 c = (Component *)sw;
 for (i=0;i<40;i++) {
  src_name[i] = src->co_name[i];
 }
 log_a_cell((Component *)c, cell, pval(sw, sw_cellog), src_name);
  ticks = USECS_TO_TICKS(pval(sw, sw_delay)->u.i);
 ev_enqueue(MY_RECEIVE, src, (Component *)sw,
    ev_now()+ticks, edgedevice_action, cell,cell->vpi,
                    (caddr_t)NULL);
  sw->total_cell++;
  return((caddr_t)sw);
}
/*****************************************/
static caddr_t
sw_my_receive(sw, cell, src)
     register RBEdgeDevice *sw;
     register Cell *cell;
     register Component *src;
{
register int  i,j, entrada,ki;
register Route_info  *rip;
register VPI  vpi;
register Component  *dest;
register Cell  *rmcell,*icell;
register Neighbor  *n;
tick_t  t1, t2;
int  connection_type;
rip = rt_lookup ((Component *)src, cell->vpi, sw->routing_table);
if (!rip) return ((caddr_t)NULL);
dest  = rip->next_link;
connection_type = rip->connection_type;
i = (int)cell->vpi;
entrada = i;
 /*************************************************************************
  *  Aqui vai ser introduzido o tempo na celula para ser possivel calcular o      *
  *  atrazo que esta vai ter dentro do Edge Device       *
  *************************************************************************/
  cell->time = ev_now();
 /*************************************************************************/
 /* locate input port for this cell */
for (j=0;j<MAXPORT;j++){
```

```
    if(sw->voport[j].to_link == src) break;
  }
   /* locate output port for this cell */
  for (i=0;i<MAXPORT;i++){
    if(sw->voport[i].to_link == dest) break;
  }
  if(i >= MAXPORT){
      fprintf(stdout, "couldn´t find virtual output port (%d); abort!\n",i);
      exit(3);
    }
    /* simply put cell in queue for purpose of enforcing speedup factor */
     for(ki=0;ki<MXVC;ki++){
           if(sw->voport[i].abr.FerRuela[ki].vpi == cell->vpi) break;
     }
     if ((connection_type == ABRCONNECTION1 ) ||
                  (connection_type == ABRCONNECTION2) ||
                  (connection_type == ABRCONNECTION3) ||
                  (connection_type == ABRCONNECTION4) ||
                  (connection_type == ATCPCONNECTION)) { /* ABR Connection */
       if ((cell->pti == 110) && (cell->u.rm.dir == 1)){
if(connection_type == ABRCONNECTION4){
if((cell->u.rm.pcr == -1.0 )&&(cell->u.rm.mcr == -1.0)){
          sw->voport[i].abr.FerRuela[ki].pcr = 0.0;
          sw->voport[i].abr.FerRuela[ki].mcr = 0.0;
          sw->voport[j].abr.pcr = 0.0;
          sw->voport[j].abr.mcr = 0.0;
          sw->voport[i].abr.FerRuela[ki].portaI = j;
          sw->voport[i].abr.FerRuela[ki].portaO = i;
          sw->voport[i].abr.FerRuela[ki].controlo = cell->u.rm.controlo;
          sw->voport[i].abr.FerRuela[ki].aplicacao = ABRCONNECTION4;
           sw->voport[i].abr.FerRuela[ki].r = cell->u.rm.r;
           sw->voport[i].abr.FerRuela[ki].b = cell->u.rm.b;
}
else {
          sw->voport[i].abr.FerRuela[ki].pcr = cell->u.rm.pcr;
          sw->voport[i].abr.FerRuela[ki].mcr = cell->u.rm.mcr;
          sw->voport[j].abr.pcr = cell->u.rm.pcr;
          sw->voport[j].abr.mcr = cell->u.rm.mcr;
          sw->voport[i].abr.FerRuela[ki].portaI = j;
          sw->voport[i].abr.FerRuela[ki].portaO = i;
          sw->voport[i].abr.FerRuela[ki].controlo = cell->u.rm.controlo;
          sw->voport[i].abr.FerRuela[ki].aplicacao = ABRCONNECTION4;
           sw->voport[i].abr.FerRuela[ki].r = cell->u.rm.r;
           sw->voport[i].abr.FerRuela[ki].b = cell->u.rm.b;
}
```

```
}
if(connection_type == ABRCONNECTION1){
        sw->voport[i].abr.FerRuela[ki].pcr = cell->u.rm.pcr;
        sw->voport[i].abr.FerRuela[ki].mcr = cell->u.rm.mcr;
        sw->voport[j].abr.pcr = cell->u.rm.pcr;
        sw->voport[j].abr.mcr = cell->u.rm.mcr;
        sw->voport[i].abr.FerRuela[ki].portaI = j;
        sw->voport[i].abr.FerRuela[ki].portaO = i;
        sw->voport[i].abr.FerRuela[ki].aplicacao = ABRCONNECTION1;
sw->voport[i].abr.controlo = cell->u.rm.controlo;
}

   }
   if ((cell->pti == 110) && (cell->u.rm.dir == 0)){
        sw->voport[j].abr.controlo = -1;
        sw->voport[j].abr.FerRuela[ki].portaControlo = -2;
sw->voport[j].abr.FerRuela[ki].ERS = sw->voport[i].abr.ERS->u.d;

    q_addt(sw->voport[i].spq, (caddr_t)cell);

   }
   /*********************************************************************/
   if ((cell->pti != 110) || ((cell->pti == 110) && (cell->u.rm.dir == 1))){
    q_addt(sw->voport[j].abrIP.ptr, (caddr_t)cell);
    pval(sw, voport[j].abrIP.iq)->u.i++;

    t2 = ev_now() - sw->voport[j].abrIP.iq->u.sample;
    if (t2 > sw->sw_log_factor->u.i) {
          log_param((Component *)sw, pval(sw, voport[j].abrIP.iq));
          sw->voport[j].abrIP.iq->u.sample = ev_now();
    }
        if(sw->voport[j].abr.burstInit == 0){
         sw->voport[j].abr.burstInit = 1;
         sw->voport[j].abr.burstInitTime = ev_now();
}
    if(sw->voport[j].abr.tokens > 0){
    if(sw->voport[j].abrIP.ptr->q_len > 0){
    icell=(Cell *)q_deq(sw->voport[j].abrIP.ptr);
    q_addt(sw->voport[j].abrOut.ptr, (caddr_t)icell);
sw->voport[j].abr.tokens--;
    pval(sw, voport[j].abrOut.dq)->u.i--;
    t2 = ev_now() - sw->voport[j].abrOut.dq->u.sample;
    if (t2 > sw->sw_log_factor->u.i) {
          log_param((Component *)sw, pval(sw, voport[j].abrOut.dq));
          sw->voport[j].abrOut.dq->u.sample = ev_now();
```

```
}
        pval(sw, voport[j].abrIP.iq)->u.i--;
        t2 = ev_now() - sw->voport[j].abrIP.iq->u.sample;
        if (t2 > sw->sw_log_factor->u.i) {
         log_param((Component *)sw, pval(sw, voport[j].abrIP.iq));
         sw->voport[j].abrIP.iq->u.sample = ev_now();
        }
        pval(sw, voport[j].abrOut.iq)->u.i++;
         t2 = ev_now() - sw->voport[j].abrOut.iq->u.sample;
         if (t2 > sw->sw_log_factor->u.i) {
          log_param((Component *)sw, pval(sw, voport[j].abrOut.iq));
          sw->voport[j].abrOut.iq->u.sample = ev_now();
         }
   }
  }
  else {
q_addt(sw->voport[j].abrIP.ptr, (caddr_t)cell);
       pval(sw, voport[j].abrIP.iq)->u.i++;
       t2 = ev_now() - sw->voport[j].abrIP.iq->u.sample;
       if (t2 > sw->sw_log_factor->u.i) {
         log_param((Component *)sw, pval(sw, voport[j].abrIP.iq));
         sw->voport[j].abrIP.iq->u.sample = ev_now();
       }
    }
   }
sw->voport[j].abr.count3++;

 } else { /* VBR connection */
     q_addt(sw->voport[i].spq, (caddr_t)cell);
 }
for(n=(Neighbor *)sw->sw_neighbors->l_head;n;n=n->n_next){
if(n->n_c == (Component *)sw->voport[i].to_link){
if(n->n_busy == FALSE){
sw_schedule_output(sw, i);
}
}
}
  return((caddr_t)sw);
}
/*****************************************/
#define PDEBUG
static caddr_t
sw_demux_spq(sw, port)
      register RBEdgeDevice *sw;
      int port;
```

```
    /*register int port;*/
{
  register int i,j,ki,kk;
  Cell *cell;
  static int count=0;
  int maxq,k,no_drop,qs,lst[MAXPORT],done,pick,match;
  char local_vci[5];
  Route_info *rip;
  int  connection_type;
  Component *src, *dest;
  tick_t t1, t2;
  int interval2;
  double macr;
  char aa;
  sw->sw_speedup = 16;
  qs = sw->voport[port].spq->q_len;
  if (no_drop=(qs>sw->sw_speedup?qs-sw->sw_speedup:0)){
#ifdef DEBUG
    fprintf(stdout,"%s (%d) too many cell in one slot--drop %d cells\n",
    sw->sw_name,ev_now(),no_drop);
    fflush(stdout);
#endif
    done=0;
    for (i=0;i<MAXPORT;i++) lst[i]=99;
    while (!done){
      for (i=0;i<no_drop;i++){
if (lst[i]==99){
  lst[i] = (int)floor(randomsim_range(0.0,(double)qs));
}
      } /* for */
      for (i=0,done=1;i<no_drop-1;i++){
if (lst[i] == lst[i+1]){
  lst[i]=99; done=0;
}
      }
    } /* while */
  }
  pick=match=0;
   while(cell=(Cell *)q_deq(sw->voport[port].spq)){
     for (i=0;i<no_drop;i++){
       if (pick==lst[i]){
match=1; break;
       }
     } /* for */
     if (match){ /* drop this cell */
```

```
        sw->num_dropped++;
        cell_free(cell);
        lst[i] = -1;
        match = 0;
#ifdef DEBUG
      fprintf(stdout,"%s (%d) cell dropped-- %d-th\n",sw->sw_name,ev_now(),pick);
      fflush(stdout);
#endif
     } else {
       rip = rt_lookup ((Component *)sw->voport[port].to_link, cell->vpi, sw->routing_table);
       if (!rip) return ((caddr_t)NULL);
       src  = rip->next_link;
       rip = rt_lookup ((Component *)src, cell->vpi, sw->routing_table);
       if (!rip) return ((caddr_t)NULL);
       dest=rip->next_link;
       for(kk=0;kk<MAXPORT;kk++)
if(sw->voport[kk].to_link == src) break;
       connection_type = rip->connection_type;
    if ((connection_type == ABRCONNECTION1) ||
        (connection_type == ABRCONNECTION4)) { /* ABR Connection */
       if ((sw->voport[port].abr.ptr->q_len < pval(sw, sw_oqsize)->u.i)
            || (pval(sw, sw_oqsize)->u.i == -1)) {
           q_addt(sw->voport[port].abr.ptr, (caddr_t)cell);
          if (sw->sw_operation->u.i == 7) {
              sw->voport[port].abr.count++;
           if ((sw->voport[port].abr.count % sw->sw_N->u.i) == 0) {
             sw->voport[port].abr.count_abr->u.d=sw->voport[port].abr.count;
              log_param((Component *)sw, pval(sw, voport[port].abr.count_abr));
              sw->voport[port].abr.rate_abr=(double)(sw->sw_N->u.i*424.0)/
                TICKS_TO_USECS(ev_now()-sw->voport[port].abr.stamp);
        sw->voport[port].vbr.rate_vbr=(double)(sw->voport[port].vbr.celulas_vbr*
                    424.0)/ TICKS_TO_USECS(ev_now()-
                    sw->voport[port].abr.stamp);
           if(sw->voport[port].vbr.celulas_vbr > 1){
            sw->IPOnOff = sw->IPOnOff + (sw->sw_TR->u.d -
(sw->voport[port].vbr.rate_vbr+sw->IPOnOff));
   } else {
            sw->IPOnOff = sw->IPOnOff1;
   }
        sw->voport[port].vbr.Rate_Vbr->u.d=sw->voport[port].vbr.rate_vbr;
        log_param((Component *)sw, pval(sw, voport[port].vbr.Rate_Vbr));
          pval(sw,sw_countvbr)->u.i=0;
          sw->voport[port].vbr.celulas_vbr=0;
            for(ki=0;ki<MXVC;ki++){
if(sw->voport[port].abr.FerRuela[ki].vpi == cell->vpi) break;
```

```
            }
    sw->voport[port].abr.FerRuela[ki].TotalCelulas++;
         log_param((Component *)sw, pval(sw, voport[port].abr.ERS));*/
         sw->voport[port].abr.stamp = ev_now();
  } /* fim do intervalo de medida */
} /* fim do 7 */
        pval(sw, voport[port].abr.iq)->u.i =(int)sw->voport[port].abr.ptr->q_len;
        t2 = ev_now() - sw->voport[port].abr.iq->u.sample;
          if (t2 > sw->sw_log_factor->u.i) {
              log_param((Component *)sw, pval(sw, voport[port].abr.iq));
              sw->voport[port].abr.iq->u.sample = ev_now();
          }
      }
      else { /* queue full, drop cells */
if (ev_now() > USECS_TO_TICKS(warmuptime)) {
  sw->num_dropped++;
  pval(sw, sw_dropped)->u.d=(double) (sw->num_dropped*100.0)/sw->total_cell;
  count = (count +1) % 10;
         }
         cell_free(cell);
        }
      }
else {
if ((sw->voport[port].vbr.ptr->q_len < pval(sw, sw_oqsize)->u.i)
            || (pval(sw, sw_oqsize)->u.i == -1)) {
          sw->voport[port].abr.vbr_cbr_cell_count += 1;
          q_addt(sw->voport[port].vbr.ptr, (caddr_t)cell);
        /* ***** Conta o numero de ceulas VBR ***** */
          sw->voport[port].vbr.celulas_vbr=sw->voport[port].vbr.celulas_vbr+1;
          sw->voport[port].vbr.count_vbr->u.d=sw->voport[port].vbr.celulas_vbr;
          log_param((Component *)sw, pval(sw, voport[port].vbr.count_vbr));
          pval(sw, sw_countvbr)->u.i = sw->sw_countvbr->u.i + 1;
          log_param((Component *)sw, pval(sw, sw_countvbr));
      } else {
sw->num_dropped++;
pval(sw, sw_dropped)->u.d=(double) (sw->num_dropped*100.0)/sw->total_cell;
        count = (count +1) % 10;
        if (count==0){
          log_param((Component *)sw, pval(sw, sw_dropped));
        }
        pval(sw, voport[port].vbr.dq)->u.i++;
           t2 = ev_now() - sw->voport[port].vbr.dq->u.sample;
          if (t2 > sw->sw_log_factor->u.i) {
          log_param((Component *)sw, pval(sw, voport[port].vbr.dq));
```

```
            sw->voport[port].vbr.dq->u.sample = ev_now();
               }
          cell_free(cell);
      } } }
    pick++;
  }
  return((caddr_t)sw);
}
/*****************************************/
static caddr_t
sw_schedule_output(sw, i)
     register RBEdgeDevice *sw;
     register int i;
{
  register Neighbor  *n,*nn;
  register Route_info  *rip;
  register int  j, k, index_port, index_vc;
  Cell  *icell;
  int  num,sel,p,s;
  static int  cnt=0;
  tick_t  t2, delay=0.0, delay_us=0.0;
  Component  *src;
  double  er_int;
int  connection_type;
int  ki;
double  a=0.0,b=0.0,c=0.0;
       for (n=(Neighbor *)sw->sw_neighbors->l_head;n;n=n->n_next) {
            if (n->n_c == (Component *) sw->voport[i].to_link) {
                 nn = n;
               n->n_busy = TRUE;
            }
       }
       if (sw->voport[i].to_link) {
  if (sw->voport[i].vbr.ptr->q_len > 0) {
            if (!(icell = (Cell *)q_deq(sw->voport[i].vbr.ptr))){
             fprintf(stdout,
              "sw_my_rec: this condition should never happen; abort!\n");
               exit(3);
          }
          sw->voport[i].vbr.iq->u.i--  ;
          t2 = ev_now() - sw->voport[i].vbr.iq->u.sample;
if (t2 > sw->sw_log_factor->u.i) {
          log_param((Component *)sw, pval(sw, voport[i].vbr.iq));
          sw->voport[i].vbr.iq->u.sample = ev_now();
}
```

```
    }
else    if (sw->voport[i].abr.ptr->q_len > 0 ) {
        if (!(icell = (Cell *)q_deq(sw->voport[i].abr.ptr))){
            fprintf(stdout, "this condition should never happen; abort!\n");
               exit(4);
        }
    pval(sw, voport[i].abr.iq)->u.i--;
        t2 = ev_now() - sw->voport[i].abr.iq->u.sample;
        if (t2 > sw->sw_log_factor->u.i) {
            log_param((Component *)sw, pval(sw, voport[i].abr.iq));
            sw->voport[i].abr.iq->u.sample = ev_now();
          }
/***********************************************
 *  Aqui vai ser calculado o atraso da celula  *
 *  dentro do Edge Device          *
 ***********************************************/
rip = rt_lookup((Component *)sw->voport[i].to_link,
                      icell->vpi, sw->routing_table);
               if (!rip) return ((caddr_t)NULL);
               for (j=0;j< MAXPORT;j++) {
                     if (sw->voport[j].to_link == rip->next_link)
                                    break;
               }
    delay = ev_now() - icell->time;
delay_us = 0.0;
    delay_us = TICKS_TO_USECS(delay);
            sw->voport[j].abrIP.dq->u.d = delay_us/1000;
    t2 = ev_now() - sw->voport[j].abrIP.dq->u.sample;
    if (t2 > sw->sw_log_factor->u.i) {
    log_param((Component *)sw, pval(sw, voport[j].abrIP.dq));
    sw->voport[j].abrIP.dq->u.sample = ev_now();
    }
    icell->time = 0.0;
        if (sw->sw_operation->u.i == 7) {
           if (icell->pti==110) {
                if (icell->u.rm.dir == 1) {
 sw->voport[i].abr.macr->u.d += (icell->u.rm.acr -
sw->voport[i].abr.macr->u.d) * 0.125;
           if(icell->u.rm.acr > sw->voport[i].abr.ACRmax->u.d){
        sw->voport[i].abr.ACRmax->u.d=icell->u.rm.acr;
log_param((Component *)sw, pval(sw, voport[i].abr.ACRmax));
 }
  t2 = ev_now() - sw->voport[i].abr.macr->u.sample;
                  if (t2 > sw->sw_log_factor->u.i) {
                    log_param((Component *)sw, pval(sw, voport[i].abr.macr));
```

```
                    sw->voport[i].abr.macr->u.sample = ev_now();
                  }
            }
            else {
                  if (icell->u.rm.dir == 0) {
                      rip = rt_lookup((Component *)sw->voport[i].to_link,
icell->vpi, sw->routing_table);
                      if (!rip) return ((caddr_t)NULL);
                      for (j=0;j< MAXPORT;j++) {
                          if (sw->voport[j].to_link == rip->next_link)
break;
                      }
                      if (icell->u.rm.NI == 0)
                           icell->u.rm.NI = sw->voport[j].abr.congested;

for(ki=0; ki<MAXPORT; ki++){
   sw->voport[ki].abr.peso=0.0;
}
for(ki=0; ki<MAXPORT; ki++){
        for(m=0; m<MXVC ;m++){
sw->voport[ki].abr.FerRuela[m].distExcesso=0;
    }
}
sw->NumVCs=0;
for(ff=0; ff<MAXPORT; ff++){
                total=0;
        for(m=0; m<MXVC ;m++){
     if(sw->voport[ff].abr.FerRuela[m].vpi !=-1){
total++;
  }
        }
        if(total>sw->NumVCs && total !=150){
sw->NumVCs=total;
    }
        }
for(ki=0; ki<MAXPORT; ki++){
                total=0;
        for(m=0; m<MXVC ;m++){
     if(sw->voport[ki].abr.FerRuela[m].vpi !=-1){
total++;
  }
        }
        if(total>1){
break;
    }
```

```
        }
totalVCs=sw->NumVCs;
Nn=totalVCs;
        rate=sw->TRaux;
 do {
totalM=0.0, NVCs=0;
PcrMcr=0.0;
for(ki=0; ki<MAXPORT; ki++){
        for(m=0; m<MXVC ;m++){
     if(sw->voport[ki].abr.FerRuela[m].vpi !=-1 &&
sw->voport[ki].abr.FerRuela[m].mcr !=0.0){
        if(sw->voport[ki].abr.FerRuela[m].distExcesso == 0){
NVCs++;
totalM=totalM+sw->voport[ki].abr.FerRuela[m].mcr;
PcrMcr=PcrMcr+sw->voport[ki].abr.FerRuela[m].pcr -
sw->voport[ki].abr.FerRuela[m].mcr;
                }
            }
         }
         }
totalP=0.0;
for(ki=0; ki<MAXPORT; ki++){
        for(m=0; m<MXVC ;m++){
     if(sw->voport[ki].abr.FerRuela[m].vpi !=-1 &&
sw->voport[ki].abr.FerRuela[m].mcr !=0.0){
        if(sw->voport[ki].abr.FerRuela[m].distExcesso == 0){
p1=sw->voport[ki].abr.FerRuela[m].mcr/totalM;
if(Nn>1){
    p2=1/(double)Nn;
    sw->voport[ki].abr.FerRuela[m].peso=(p1+p2)/2;
        }
        else {
   sw->voport[ki].abr.FerRuela[m].peso=1.0;
        }
totalP=totalP+sw->voport[ki].abr.FerRuela[m].peso;
                }
            }
         }
         }
for(km=0; km<MAXPORT; km++){
        for(m=0; m<MXVC ;m++){
     if(sw->voport[km].abr.FerRuela[m].vpi !=-1 &&
sw->voport[km].abr.FerRuela[m].mcr !=0.0){
        if(sw->voport[km].abr.FerRuela[m].distExcesso == 0){
sw->voport[km].abr.FerRuela[m].ERQ=sw->voport[km].abr.FerRuela[m].mcr+
```

```
 (rate-totalM)* sw->voport[km].abr.FerRuela[m].peso;
     }
  }
    }
}
countt=0;
banda=0.0;
for(km=0; km<MAXPORT; km++){
        for(m=0; m<MXVC ;m++){
     if(sw->voport[km].abr.FerRuela[m].vpi !=-1 &&
sw->voport[km].abr.FerRuela[m].mcr !=0.0){
if(sw->voport[km].abr.FerRuela[m].ERQ>sw->voport[km].abr.FerRuela[m].pcr){
banda = banda +sw->voport[km].abr.FerRuela[m].pcr;
sw->voport[km].abr.FerRuela[m].distExcesso=1;
        sw->voport[km].abr.FerRuela[m].ERQ=sw->voport[km].abr.FerRuela[m].pcr;
}
else {
countt++;
}
  }
    }
}
rate=rate-banda;
          uu++;
Nn=0;
for(ki=0; ki<MAXPORT; ki++){
        for(m=0; m<MXVC ;m++){
     if(sw->voport[ki].abr.FerRuela[m].vpi !=-1 &&
sw->voport[ki].abr.FerRuela[m].mcr !=0.0){
        if(sw->voport[ki].abr.FerRuela[m].distExcesso == 0){
Nn++;
                }
            }
         }
         }
if(Nn<=1) Nn=1;
} while (countt < totalVCs );
for(ki=0; ki<MAXPORT; ki++){
                total=0;
        for(m=0; m<MXVC ;m++){
     if(sw->voport[ki].abr.FerRuela[m].vpi !=-1){
total++;
  }
        }
        if(total>1){
```

```
break;
    }
        }
if (icell->u.rm.er > sw->voport[i].abr.FerRuela[icell->vpi].ERQ)
      icell->u.rm.er = sw->voport[i].abr.FerRuela[icell->vpi].ERQ;
t2 = ev_now() - sw->voport[j].abr.ERS->u.sample;
         if (t2 > sw->sw_log_factor->u.i) {
             log_param((Component *)sw, pval(sw, voport[j].abr.ERS));
             sw->voport[j].abr.ERS->u.sample = ev_now();
          }
          } /* fim do DIR = 0*/
        }
            }
          }
          ev_enqueue (EV_RECEIVE, (Component *)sw, sw->voport[i].to_link,
                      ev_now(), sw->voport[i].to_link->co_action, icell, icell->vpi, (caddr_t)NULL);
    }
  return((caddr_t)sw);
}
/****************************************/
static caddr_t
sw_proc_slot_time(sw)
     register RBEdgeDevice *sw;
{
  register Neighbor  *n;
  register int  j, Flag,ki;
  int  i;
  Cell  *cell;
  int  ci;
     for (i=0;i<MAXPORT; i++) {
       if (sw->voport[i].to_link) {
          sw_demux_spq(sw, i);
       }
     }
  for (i=0;i<MAXPORT;i++){
    if (sw->voport[i].to_link){
       if (((sw->voport[i].abr.ptr->q_len > 0) ||
            (sw->voport[i].vbr.ptr->q_len > 0)) &&
(sw->voport[i].link_busy == FALSE )) {
            sw->voport[i].link_busy = TRUE;
            sw_schedule_output(sw, i);
      }
    }
  }
  return((caddr_t)sw);
```

```
}
/****************************************/
static caddr_t
sw_ready(sw, src)
     register RBEdgeDevice *sw;
     register Component *src;
{
     register Neighbor *n;
  int i;
 for (n=(Neighbor *)sw->sw_neighbors->l_head;n;n=n->n_next) {
       if ((n->n_c == src) && (n->n_busy == TRUE))
          n->n_busy = FALSE ;
 }
for (i=0;i<MAXPORT; i++) {
    if (sw->voport[i].to_link == src) break;
}
if (sw->voport[i].link_busy == TRUE ) {
    sw->voport[i].link_busy = FALSE;
    if (sw->voport[i].to_link){
         if ((sw->voport[i].vbr.ptr->q_len > 0)
            || (sw->voport[i].abr.ptr->q_len > 0)) {
     sw->voport[i].link_busy = TRUE;
     sw_schedule_output (sw, i);
           }
       }
   }
  return((caddr_t)sw);
}
/****************************************/
static caddr_t
sw_averaging_int_time(sw)
     register RBEdgeDevice *sw;
{
  int i, j,p;
  tick_t t2;
  int num, index_vc;
  int k;
  double tm,tp;
#ifdef DEBUG
 dbg_write(debug_log, DBG_INFO, (Component *)NULL, "Switch %s AI expires", sw->sw_name);
#endif
  for (i=0;i<MAXPORT;i++){
    if (sw->voport[i].to_link){
#ifdef DEBUG
      dbg_write(debug_log, DBG_INFO, (Component *)NULL, "\tPort to link %s:", ((Component *) sw->voport[i].to_li
```

```
#endif
      if ( sw->sw_operation->u.i == 7 ) {
            sw->TotalMCR = 0.0;
            sw->TotalPCR = 0.0;
            tp=tm=0.0;
       for(j=0;j<MXVC;j++){
  if(sw->voport[i].abr.FerRuela[j].vpi !=-1){
p=sw->voport[i].abr.FerRuela[j].porta0;
if(p!=-1 && sw->voport[i].abr.FerRuela[j].controlo == 0){
    tp=tp + sw->voport[i].abr.FerRuela[j].pcr;
    tm=tm + sw->voport[i].abr.FerRuela[j].mcr;
    sw->E10 = (sw->sw_AI->u.d * sw->voport[i].abr.FerRuela[j].mcr)/424.0;
    sw->voport[p].abr.Fr->u.d = sw->voport[p].abr.Fr->u.d + sw->E10 - (int)sw->E10;
    sw->voport[p].abr.w1->u.i = (int)sw->E10;
    log_param((Component *)sw, pval(sw, voport[p].abr.w1));
    sw->E11 = (sw->sw_AI->u.d * (sw->voport[i].abr.FerRuela[j].ERS -
sw->voport[i].abr.FerRuela[j].mcr))/424.0;
    sw->voport[p].abr.Fr->u.d = sw->voport[p].abr.Fr->u.d + sw->E11 - (int)sw->E11;
    sw->voport[p].abr.w2->u.i = (int)sw->E11;
    log_param((Component *)sw, pval(sw, voport[p].abr.w2));
}
  }
      }
           if((tp==0.0) && (tm==0.0)){
        sw->TotalMCR = 1.0;
        sw->TotalPCR = 1.0;
           }
     else {
        sw->TotalMCR = tm;
        sw->TotalPCR = tp;
           }

     for(j=0;j<MXVC;j++){
if(sw->voport[i].abr.FerRuela[j].vpi != -1){
                  sw->voport[i].abr.FerRuela[j].TotalCelulas = 0;
              }
           }
      }
    } /* end if port used */
  } /* end for i */
  return((caddr_t)sw);
}
/***********************************/
static caddr_t
sw_cell_scheduling(sw)
```

```
     register RBEdgeDevice *sw;
{
     int  fla = 0,Zmax=-999,Zmin=999,FzMin = -1,delta=0,i, j, m, w,total=0, total1=0;
     int Fonte=-1, FonteOld = -1, FonteDmax = -1, NQ=0;
     int ff=0,kk1=0,Fonte1=-1, Min1=99999, Min2=99999, totalZ=0, Max=0 ,Max2=0,
totalX=1,MaxQ=0, MinQ = 99999;
     tick_t  cell_time, delay=0.0;
     Cell  *cell, *jcell;
     Neighbor  *n,*n2;
     tick_t  t2, ddly=0.0;
     Component  *dest;
     Route_info  *rip;
     register int  kk, jj;
     double  Tmax = 0.0, T = 0.0,q=0.0,b=0.0,soma = 0.0,rate=0.0, t11=0.0, t22=0.0,
t33=0.0, t44=0.0,Max1=0.0, Min=99999.0;
     double  inter=0.0,totalSCR=0.0,delay_us=0.0, totalQ=0.0, debito=0.0,
atraso=0.0, limiar=0.0;
     double  dd=0.0,aa=0.0,bb=0.0;
        NQ=0;
kk1=0;
for(i=0; i<MAXPORT; i++){
          total=0;
   for(m=0; m<MXVC ;m++){
if(sw->voport[i].abr.FerRuela[m].vpi !=-1){
total++;
}

   }
   if(total==1){
          if (sw->voport[i].abrOut.ptr) {
             if (sw->voport[i].abrOut.ptr->q_len==0) {
kk1++;
}
}
   }
}
if(kk1==0){
sw->interval1= sw->interval;
}
for(i=0; i<MAXPORT; i++){
          total=0;
   for(m=0; m<MXVC ;m++){
if(sw->voport[i].abr.FerRuela[m].vpi !=-1){
total++;
}
```

```
   }
   if(total==1){
           if (sw->voport[i].abrOut.ptr) {
              if (sw->voport[i].abrOut.ptr->q_len>0) {
sw->voport[i].abr.x = sw->voport[i].abr.b -
                 (sw->voport[i].abr.tokens + sw->voport[i].abrOut.ptr->q_len);
totalSCR = totalSCR + pval(sw, voport[i].abr.SCR)->u.d;
                              totalZ = totalZ + sw->voport[i].abr.tokens;
                              totalX = totalX + sw->voport[i].abr.x;
                              totalQ = totalQ + sw->voport[i].abrOut.ptr->q_len;
/* calculo do tempo de espera maximo */
T = 0.0;
if(sw->voport[i].abr.delay > 1.0)
T = TICKS_TO_USECS(ev_now() - sw->voport[i].abr.delay);
if(T >= Tmax) {
Tmax = T;
FonteDmax = i;
}
/*********************************************/
NQ++; /* numero de buffers com celulas */
}
}
   }
}
/* so faz um incremento de cada vez que vem ca´ para que nao seja injusto */
if(NQ > 0)
sw->timeInterval = sw->timeInterval + sw->cellSchedulingTime;
else{
sw->tFlag=0;
sw->creditsMin=10000;
for(i=0; i<MAXPORT; i++){
            sw->voport[i].abr.creditsAcum=0;
}
}
for(i=0; i<MAXPORT; i++){
           total=0;
   for(m=0; m<MXVC ;m++){
if(sw->voport[i].abr.FerRuela[m].vpi !=-1){
total++;
}
   }
   if(total==1){
           if (sw->voport[i].abrOut.ptr) {
              if (sw->voport[i].abrOut.ptr->q_len>0) {
```

```
T = 0.0;
if(sw->voport[i].abr.delay > 1.0)
T = TICKS_TO_USECS(ev_now() - sw->voport[i].abr.delay);
    }
                              /*************************************************************
                               *      Armazenar o valor de X                               *
                               *************************************************************/
   pval(sw, voport[i].abr.dq)->u.d=sw->voport[i].abr.x;
          t2 = ev_now() - sw->voport[i].abr.dq->u.sample;
          if (t2 > sw->sw_log_factor->u.i) {
       log_param((Component *)sw, pval(sw, voport[i].abr.dq));
       sw->voport[i].abr.dq->u.sample = ev_now();
      }
   pval(sw, voport[i].abr.k)->u.d++;
          t2 = ev_now() - sw->voport[i].abr.k->u.sample;
          if (t2 > sw->sw_log_factor->u.i) {
       log_param((Component *)sw, pval(sw, voport[i].abr.k));
       sw->voport[i].abr.k->u.sample = ev_now();
      }
                              /*************************************************************
                               *      Calcular e armazenar o valor do atraso               *
                               *************************************************************/
if(sw->voport[i].vbr.dq->u.d > sw->voport[i].abr.Rate_Abr->u.d
    && sw->voport[i].vbr.dq->u.d != 0.0 && sw->voport[i].abr.Rate_Abr->u.d != 0.0){
atraso = (sw->voport[i].abrOut.ptr->q_len * 424.0) /
                    (sw->voport[i].vbr.dq->u.d - sw->voport[i].abr.Rate_Abr->u.d);
        sw->voport[i].vbr.iq->u.d = atraso/1000;
                  log_param((Component *)sw, pval(sw, voport[i].vbr.iq));
}
else {
atraso = 0.0;
        sw->voport[i].vbr.iq->u.d = atraso;
                  log_param((Component *)sw, pval(sw, voport[i].vbr.iq));
}
/**** Calcular qual a fonte a escalonar *******/
if(sw->voport[i].abr.x>=0) {
if(sw->voport[i].abr.tokens > Max && sw->voport[i].abr.credits <= sw->creditsMin &&
    sw->voport[i].abr.flagA == 0) {
Max = sw->voport[i].abr.tokens;
Fonte = i;
}
else {
/* verificar o periodo de tempo */
if(sw->voport[i].abr.flagA == 0 || sw->voport[i].abr.credits > sw->creditsMin) {
if(sw->voport[i].abr.credits > sw->creditsMin) {
```

```
sw->voport[i].abr.flagA = 1;
Fonte = FonteDmax;
sw->voport[FonteDmax].abr.flagA = 0;
}
}
}
}
else {
sw->voport[i].abr.flagA = 0;
Fonte = i;
sw->flagX=1;
}
if(NQ==1){
sw->tFlag = 0;
sw->flag = Fonte;

}
else{
if(Fonte != sw->flag && Fonte != -1){
sw->voport[sw->flag].abr.flagA = 1;
aa = (double)sw->voport[sw->flag].abr.tokens;
bb = (double)sw->voport[sw->flag].abr.b;
dd=0.0;
dd = aa / bb;
if(sw->tFlag == 0){
sw->tFlag = 1;
if(sw->voport[sw->flag].abr.creditsAcum == 0){
sw->creditsMin = sw->creditsMin1;
}
else {
     if(dd < 0.5)
sw->intCredits = pval(sw, voport[sw->flag].abr.MBS)->u.i -
sw->voport[sw->flag].abr.credits;
     else
sw->intCredits = sw->voport[sw->flag].abr.credits;
}
}
else {
if(sw->flagX==1){
sw->intCredits = sw->intCredits;
sw->flagX=0;
}
else {
sw->intCredits = sw->voport[sw->flag].abr.credits;
}
```

```
}
if((sw->creditsMin > sw->intCredits) &&(sw->voport[sw->flag].abr.creditsAcum!=0))
sw->creditsMin = sw->intCredits;
sw->voport[sw->flag].abr.credits=0;
sw->flag = Fonte;
}
}/* else NQ > 1 */
sw->interval=sw->interval1;
      }
else {
         sw->voport[i].abr.flagA = 0;
         sw->voport[i].abr.credits = 0;
}
     }
  } else {
   if(total ==2) { /* fontes= 1 2 3 4 */

w=i;
           }
}
     }
/* Fontes com trafego mas nenhuma foi seleccionada */
if(Fonte == -1 && NQ != 0){
Fonte = FonteDmax;
sw->voport[FonteDmax].abr.flagA = 0;
sw->flag = Fonte;
sw->timeInterval = 0.0;
sw->voport[FonteDmax].abr.credits = 0;
}
        if (sw->voport[Fonte].abrOut.ptr) {
        if (sw->voport[Fonte].abrOut.ptr->q_len>0) {
/* armazenar o Z, X e Q  acumulado */
   pval(sw, voport[w].abr.w1)->u.i=totalZ;
           t2 = ev_now() - sw->voport[w].abr.w1->u.sample;
           if (t2 > sw->sw_log_factor->u.i) {
       log_param((Component *)sw, pval(sw, voport[w].abr.w1));
       sw->voport[w].abr.w1->u.sample = ev_now();
   }
pval(sw, voport[w].abr.w2)->u.i=totalX;
        t2 = ev_now() - sw->voport[w].abr.w2->u.sample;
        if (t2 > sw->sw_log_factor->u.i) {
    log_param((Component *)sw, pval(sw, voport[w].abr.w2));
    sw->voport[w].abr.w2->u.sample = ev_now();
   }
pval(sw, voport[w].abr.Fr)->u.d=totalQ;
```

```
        t2 = ev_now() - sw->voport[w].abr.Fr->u.sample;
        if (t2 > sw->sw_log_factor->u.i) {
     log_param((Component *)sw, pval(sw, voport[w].abr.Fr));
     sw->voport[w].abr.Fr->u.sample = ev_now();
    }
pval(sw, voport[Fonte].abr.k)->u.d=0.0;
        t2 = ev_now() - sw->voport[Fonte].abr.k->u.sample;
        if (t2 > sw->sw_log_factor->u.i) {
     log_param((Component *)sw, pval(sw, voport[Fonte].abr.k));
      sw->voport[Fonte].abr.k->u.sample = ev_now();
     }
/* Armazenar o tempo em que a fonte foi escalonada */
sw->voport[Fonte].abr.delay = ev_now();
  /*******************************************************/
                  sw->voport[Fonte].abr.credits++;
                  sw->voport[Fonte].abr.creditsAcum++;
                  sw->voport[Fonte].abr.count1++;
                  sw->voport[Fonte].abrOut.state= ACTIVE;
   cell = (Cell *)q_deq(sw->voport[Fonte].abrOut.ptr);
   rip = rt_lookup((Component *)sw->voport[Fonte].to_link, cell->vpi, sw->routing_table);
   if(!rip) return((caddr_t)NULL);
   dest = rip->next_link;
   for(kk = 0; kk <MAXPORT; kk++) {
if(sw->voport[kk].to_link == dest) break;
   }
   for(jj = 0; jj <MXVC; jj++) {
if(sw->voport[kk].abr.FerRuela[jj].vpi == cell->vpi) break;
   }
   if((sw->voport[kk].abr.ptr->q_len < sw->sw_oqsize->u.i) ||
(sw->sw_oqsize->u.i == -1)){
                  q_addt(sw->voport[kk].spq,(caddr_t)cell);
    pval(sw, voport[Fonte].abrOut.iq)->u.i--;
    t2 = ev_now() - sw->voport[Fonte].abrOut.iq->u.sample;
    if (t2 > sw->sw_log_factor->u.i) {
    log_param((Component *)sw, pval(sw, voport[Fonte].abrOut.iq));
    sw->voport[Fonte].abrOut.iq->u.sample = ev_now();
    }
              }
         }
   else {
            if (sw->voport[Fonte].abrOut.ptr)
                sw->voport[Fonte].abrOut.state = IDLE;
         }
      }
return ((caddr_t)sw);
```

```
}
/****************************************/
static caddr_t
sw_comput_period_on(sw)
     register RBEdgeDevice *sw;
{
  int i, j,p,p1;
  tick_t ticks;
  int vci, port;
  VPI vpi = sw->vpi;
  if(sw->period_on_initial == 1){
sw->period_on_initial = 0;
ticks=USECS_TO_TICKS(4000);
   for (i=0;i<MAXPORT;i++){
     if (sw->voport[i].to_link){
        for(j=0;j<MXVC;j++){
   if(sw->voport[i].abr.FerRuela[j].vpi !=-1){
p=sw->voport[i].abr.FerRuela[j].portaI;
p1=sw->voport[i].abr.FerRuela[j].porta0;
if(p!=-1){
sw->voport[p].abr.on = 1;
             ev_enqueue(MY_DONE_ON, (Component *)sw,
                                    (Component *)sw, ev_now()+ticks,
                                    edgedevice_action,
                                    (Cell *)NULL, p, (caddr_t)NULL);
}
   }
       }
}
}
   }
   else {
sw->voport[vpi].abr.on = 1;
ticks=USECS_TO_TICKS(sw->voport[vpi].abr.Ton);
    ev_enqueue(MY_DONE_ON, (Component *)sw, (Component *)sw,
                  ev_now()+ticks, edgedevice_action, (Cell *)NULL, vpi, (caddr_t)NULL);
   }
  return((caddr_t)sw);
}
/****************************************/
static caddr_t
sw_schedule_off_period(sw)
     register RBEdgeDevice *sw;
{
  int i, j;
```

```
tick_t ticks;
int vci, port;
VPI vpi = sw->vpi;
if(sw->voport[vpi].abr.pcr != 0.0 && sw->voport[vpi].abr.ERS->u.d !=0.0) {
       sw->voport[vpi].abr.b = (int)(500* (sw->voport[vpi].abr.pcr -
 sw->voport[vpi].abr.ERS->u.d))/ sw->voport[vpi].abr.pcr;
       sw->voport[vpi].abr.Toff = (int)((sw->voport[vpi].abr.b*424.0)
                                        /sw->voport[vpi].abr.ERS->u.d);
       ticks=USECS_TO_TICKS(sw->voport[vpi].abr.Toff);
       sw->voport[vpi].abr.on = 0; /* Passagem ao estado OFF */
       ev_enqueue(MY_COMPUT_PERIOD_ON, (Component *)sw, (Component *)sw,
                 ev_now()+ticks, edgedevice_action,
                 (Cell *)NULL, vpi, (caddr_t)NULL);
  }
  return((caddr_t)sw);
}
/*****************************************/
static caddr_t
sw_comput_tokens(sw,aa)
     register RBEdgeDevice *sw;
     VPI aa;
{
  int  i, j, kk,jj,p, porta;
  tick_t  ticks, t2, cell_time;
  int  vci, port;
  Cell  *icell;
  register  Route_info *rip;
  register  Neighbor *n;
  register  Component *dest;
  VPI  vpi=sw->vpi;
  double tempo=0.0;
   if(sw->voport[aa].abr.tokens < sw->voport[aa].abr.b) {
         sw->voport[aa].abr.tokens++;
    pval(sw, voport[aa].abrOut.dq)->u.i++;
         log_param((Component *)sw, pval(sw, voport[aa].abrOut.dq));
}
   /* Se existirem tokens disponiveis e celulas na fila de entrada entao enviar a
celula para a fila de saida */
   if(sw->voport[aa].abr.tokens > 0) {
   if(sw->voport[aa].abrIP.ptr->q_len > 0) {
   icell = (Cell *)q_deq(sw->voport[aa].abrIP.ptr);
           sw->voport[aa].abr.tokens--;
       pval(sw, voport[aa].abrOut.dq)->u.i--;
       t2 = ev_now() - sw->voport[aa].abrOut.dq->u.sample;
       if (t2 > sw->sw_log_factor->u.i) {
```

```
        log_param((Component *)sw, pval(sw, voport[aa].abrOut.dq));
        sw->voport[aa].abrOut.dq->u.sample = ev_now();
}
q_addt(sw->voport[aa].abrOut.ptr,(caddr_t)icell);
    pval(sw, voport[aa].abrIP.iq)->u.i--;
    t2 = ev_now() - sw->voport[aa].abrIP.iq->u.sample;
    if (t2 > sw->sw_log_factor->u.i) {
 log_param((Component *)sw, pval(sw, voport[aa].abrIP.iq));
    sw->voport[aa].abrIP.iq->u.sample = ev_now();
       }
         pval(sw, voport[aa].abrOut.iq)->u.i++;
       t2 = ev_now() - sw->voport[aa].abrOut.iq->u.sample;
       if (t2 > sw->sw_log_factor->u.i) {
    log_param((Component *)sw, pval(sw, voport[aa].abrOut.iq));
    sw->voport[aa].abrOut.iq->u.sample = ev_now();
       }
   rip = rt_lookup((Component *)sw->voport[aa].to_link, icell->vpi, sw->routing_table);
   if(!rip) return((caddr_t)NULL);
   dest = rip->next_link;
   for(kk = 0; kk <MAXPORT; kk++) {
if(sw->voport[kk].to_link == dest) break;
   }
}
  return((caddr_t)sw);
}
/****************************************/
static caddr_t
sw_tokenBucket_init(sw)
     register RBEdgeDevice *sw;
{
  int  i,total,m;
  tick_t  ticks, t2, cell_time;
  double        tempo=0.0;
for(i=0;i<3; i++){
              total=0;
for(m=0; m<MXVC ;m++){
if(sw->voport[i].abr.FerRuela[m].vpi !=-1){
total++;
}
}
if(sw->voport[i].abr.SCR->u.d != 0.0){
    sw->voport[i].abr.b = ((pval(sw, voport[i].abr.MBS)->u.i *
                (pval(sw, voport[i].abr.PCR)->u.d - pval(sw, voport[i].abr.SCR)->u.d))
                           / pval(sw, voport[i].abr.PCR)->u.d) + 1;
```

```
    sw->voport[i].abr.tokens = sw->voport[i].abr.b; /* token cheio inicialmente */
            pval(sw, voport[i].abrOut.dq)->u.i=sw->voport[i].abr.tokens;
          log_param((Component *)sw, pval(sw, voport[i].abrOut.dq));
sw->voport[i].abr.burstTransmissionTime =
(pval(sw, voport[i].abr.MBS)->u.i * 424.0)/
                                    pval(sw, voport[i].abr.PCR)->u.d;
            cell_time = USECS_TO_TICKS(424.0/sw->voport[i].abr.SCR->u.d);
            tempo = 424.0/sw->voport[i].abr.SCR->u.d;
              ev_enqueue(MY_COMPUT_TOKENS, (Component *)sw,
                         (Component *)sw, ev_now()+cell_time,
                         edgedevice_action, (Cell *)NULL, i,
                         (caddr_t)NULL);
}
}
  return((caddr_t)sw);
}
/****************************************/
static caddr_t
sw_comput_rate(sw)
     register RBEdgeDevice *sw;
{
  int  i,total,m, diferenca=0;
  tick_t  ticks, t2, cell_time;
  double        rate=0.0, rateIN=0.0,tempo1=0.0;
for(i=0;i<3; i++){
tempo1 = TICKS_TO_USECS(ev_now());
                total=0;
for(m=0; m<MXVC ;m++){
if(sw->voport[i].abr.FerRuela[m].vpi !=-1){
total++;
}
}
if(total == 1 && sw->voport[i].abr.count1 >0){
rate=0.0;
rate = (double)((sw->voport[i].abr.count1 * 424.0)/(1000));
pval(sw, voport[i].abr.Rate_Abr)->u.d= pval(sw, voport[i].abr.Rate_Abr)->u.d+rate;
        log_param((Component *)sw, pval(sw, voport[i].abr.Rate_Abr));
sw->voport[i].abr.count1 = 0;
}
if(total == 1 && sw->voport[i].abr.count3 >0){
rateIN = (double)((sw->voport[i].abr.count3 * 424.0)/(1000));
pval(sw, voport[i].vbr.dq)->u.d=rateIN;
        log_param((Component *)sw, pval(sw, voport[i].vbr.dq));
sw->voport[i].abr.count3 = 0;
} else {
```

```
if(total == 1 && sw->voport[i].abr.count3 <= 0){
sw->voport[i].vbr.dq->u.d = 0.0;
}

} /* else */
        }
        cell_time = USECS_TO_TICKS(1000);
ev_enqueue(MY_RATE, (Component *)sw, (Component *)sw, ev_now()+cell_time,
                edgedevice_action, (Cell *)NULL, NULL, (caddr_t)NULL);
  return((caddr_t)sw);
}
```

Apêndice C

Avaliação de algoritmos de escalonamento

Neste anexo é apresentada uma avaliação, com base em resultados de simulação que podem ser considerados como referências para outras disciplinas a usar nos equipamentos periféricos da rede. Como será observado durante a discussão dos resultados obtidos, por um lado a disciplina de serviço FIFO não apresenta nenhuma surpresa, uma vez que esta disciplina não foi concebida para dar qualquer tipo de garantias e, por outro lado a ideia generalizada do potencial da disciplina de serviço WFQ fica aquém do esperado, tendo em conta os objectivos previamente estabelecidos. É, no entanto, necessário referir que esta disciplina garante a distribuição da largura de banda e os limites de atraso máximo entre as fontes em competição.

Algumas notas introdutórias são apresentadas na primeira secção relativamente às disciplinas de serviço utilizadas nos testes de simulação. Na secção seguinte são apresentados os detalhes de implementação, os resultados de simulação sobre os seis cenários previamente definidos e algumas conclusões para a disciplina de serviço FIFO; a discussão procurou identificar os problemas associados. Na terceira secção são mostrados os detalhes de implementação e os resultados de simulação obtidos sobre os mesmos cenários, para a disciplina de serviço WFQ.

C.1 Introdução

A existência de um conjunto diversificado de disciplinas de serviço, como apresentado no Captítulo 4, mostra a preocupação existente em relação ao problema da garantia de serviço. Em muitos resultados de simulação apresentados na literatura apenas se considera o caso de redes com tecnologia IP ignorando-se outras tecnologias combinadas com o IP. No entanto quando se agregam fluxos IP com necessidades de QoS em redes ATM, especialmente em VCs ABR, é necessário observar que a existência de tráfego VBR e/ou CBR significa que a banda disponível para tráfego ABR varia, pelo que é necessário testar as disciplinas de serviço propostas nessas condições. No presente anexo discutem-se duas disciplinas de serviço implementadas nos EDs.

Uma das soluções testadas em relação à distribuição da largura de banda e dos atrasos é o tradicional FIFO. Este disciplina baseia-se essencialmente em servir células pela respectiva ordem de chegada e desde que a soma dos débitos médios não exceda a capacidade disponível são assegurados os atrasos dentro dos limites, bem como a distribuição da largura de banda sem considerar factores como pesos de atribuição ou classes, independentemente da preocupação em distribuir de forma justa os atrasos e a ocupação das filas de espera à entrada dos EDs para cada um dos fluxos IP.

A segunda disciplina de serviço em estudo (WFQ) é uma das mais conhecidas na literatura. O modo de funcionamento desta disciplina é interessante do ponto de vista teórico porque de forma simples, como conceito, consegue dar garantias, isto é, a atribuição de tempos de chegada às células e o cálculo dos tempos de partida com base no débito atribuído e no tamanho da célula permite fornecer garantias necessárias quanto à largura de banda distribuída, bem como aos atrasos associados.

Os resultados obtidos com a disciplina de serviço WFQ serão analisados ao longo desta secção e será interessante observar que à medida que a complexidade das configurações vai aumentando a reacção da disciplina de serviço tem um comportamento bastante peculiar porque continua a garantir os débitos e os atrasos dentro do intervalo definido, mas a equidade na distribuição dos atrasos entre as fontes em competição desaparece progressivamente.

C.2 Disciplina de serviço FIFO

Nesta secção serão apresentados detalhes de implementação, os resultados de simulação com a respectiva discussão para os seis cenários e extraídas algumas conclusões para a disciplina de serviço FIFO.

C.2.1 Detalhes de implementação

Antes de passar propriamente para a análise dos resultados é necessário realizar algumas considerações relativamente ao processo de implementação desta disciplina no simulador. Os resultados obtidos com a disciplina de serviço FIFO basearam-se na sua implementação no simulador descrito no Capítulo 3. Esta implmentação não apresenta qualquer tipo de desafio porque apenas é necessário receber as células dos vários fluxos de entrada e transferi-los para a fila de saída. Para a obtenção da ocupação das filas de entrada basta associar um contador a cada fila de entrada que é incrementado quando a célula chega e decrementado quando sai. Quanto ao atraso o processo é praticamente o mesmo, com a diferença que às células é associado o tempo corrente de simulação no momento em que chegam e na saída é verificado o tempo de entrada e calculado o atraso sofrido dentro do equipamento.

C.2.2 Resultados de simulação

Nesta secção serão apresentados os resultados obtidos através da simulação com recurso a seis cenários (ver secção 5.5) que servirão como base para a discussão relativamente aos méritos e problemas da disciplina de serviço FIFO em diferentes configurações.

Cenário 1

Os resultados apresentados na Figura C.1 mostram o desempenho desta disciplina de serviço. A primeira conclusão que se pode extrair é o bom desempenho em termos gerais, isto é, a ocupação das filas de espera de entrada e a distribuição dos débitos encontram-se dentro do esperado. Contudo, é possível observar que, apesar da configuração ser simples, apresenta desde logo uma falta de equidade entre as duas fontes em competição que se pode resumir da seguinte forma: a ocupação da fila de entrada da segunda fonte é superior à da primeira, o que se reflete nos atrasos observados. Como conclusão destes

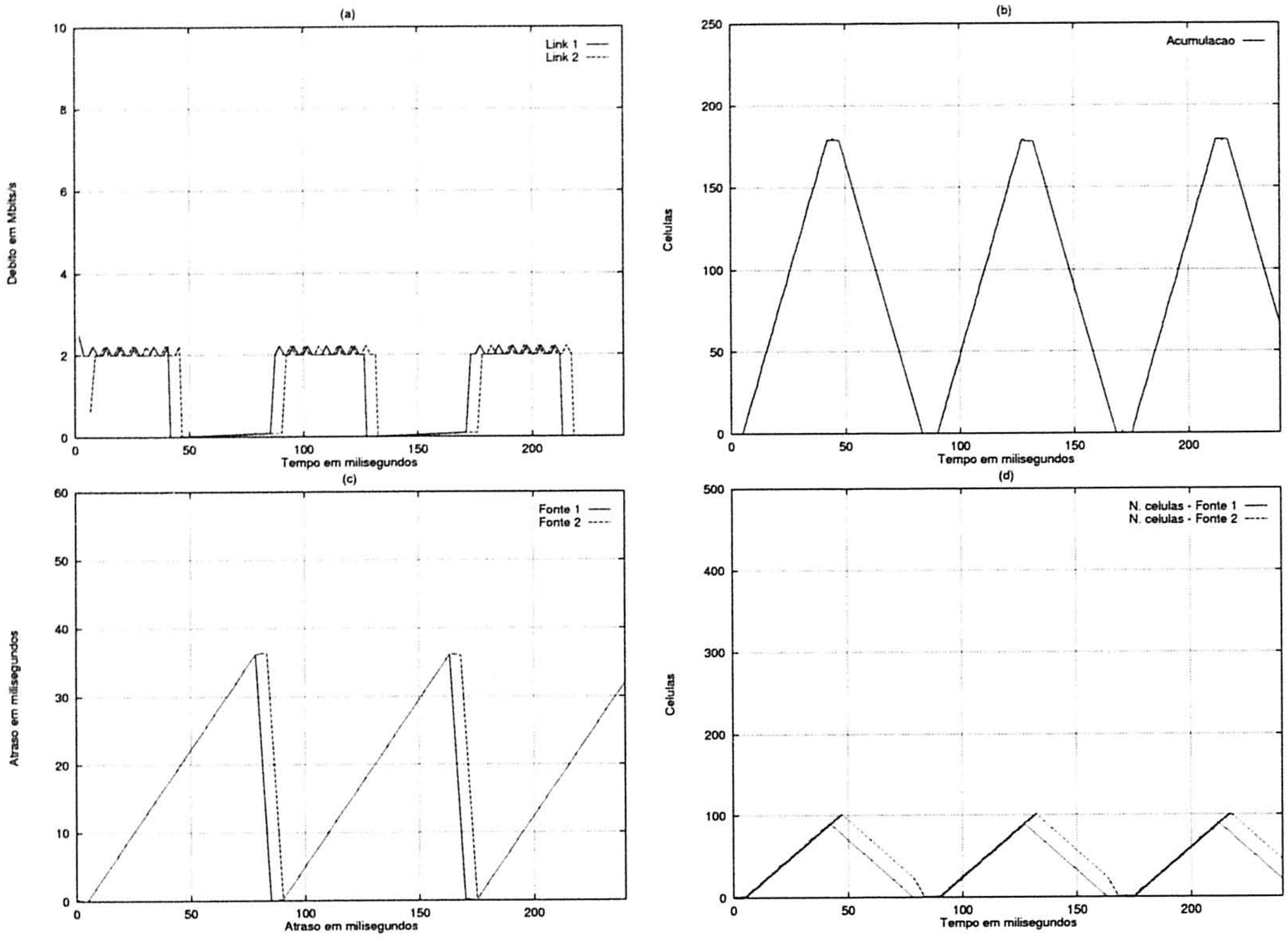

Figura C.1: Resultados obtidos com a disciplina de serviço FIFO no Cenário 1.

resultados a fonte que inicia a sua actividade mais cedo beneficia em relação à que começa mais tarde.

Cenário 2

As conclusões que se podem tirar destes resultados (ver Figura C.2) mostram que a disciplina de serviço realiza a sua função sem criar qualquer tipo de problema, isto é, durante o tempo em que apenas uma fonte estava em actividade, tanto a ocupação da fila de entrada, bem como os atrasos são nulos, mas no instante em que a segunda fonte começa a transmitir o número de células que restam ainda do *burst* da primeira são muito poucos o que levou a que não existisse qualquer situação anómala.

Cenário 3

Os resultados apresentados na Figura C.3 são os esperados relativamente às ocupações e a utilização dos recursos. Quanto à ocupação da filas de espera das fontes pode-se verificar que, quando comparado com os resultados da Figura C.1, a ocupação observada na fila de espera de entrada referente à Fonte 1 apresenta sensivelmente uma menor

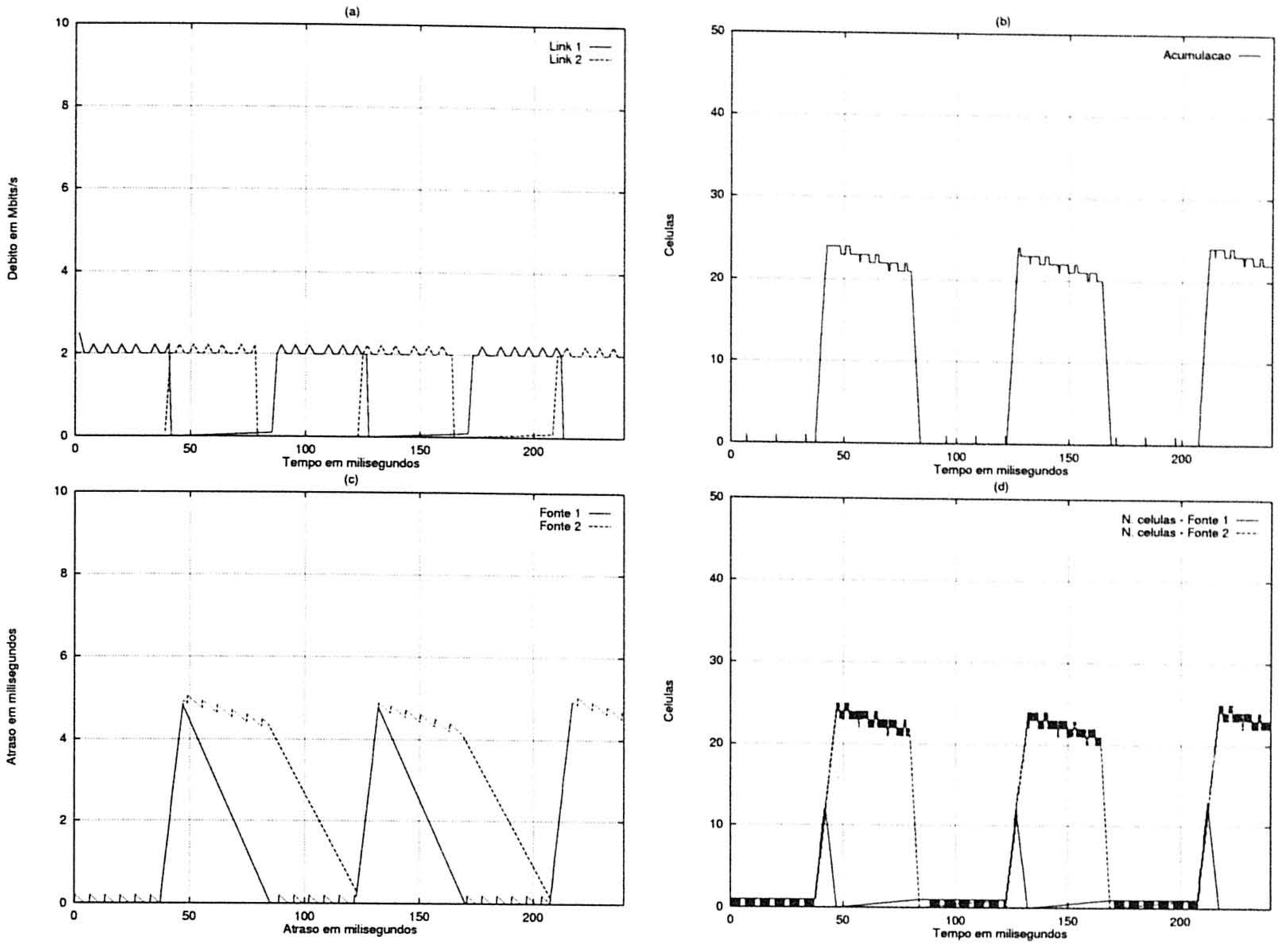

Figura C.2: Resultados obtidos com a disciplina de serviço FIFO no Cenário 2.

ocupação, mas com atrasos superiores devido à existência das outras duas fontes também em actividade. Mais uma vez se percebe que a fonte que inicia a sua actividade mais cedo é beneficiada relativamente às outras que iniciam mais tarde, mesmo que a soma dos débitos seja superior.

Cenário 4

As conclusões que se podem tirar relativamente aos resultados apresentados na Figura C.4 são simples porque a distribuição é realizada com base no débito de transmissão das fontes e não na preocupação de garantir o débito acordado inicialmente. Esta falta de garantias verifica-se nas ocupações das filas de entrada e consequentemente na distribuição dos atrasos, uma vez que apesar de uma transmitir ao triplo do débito da outra os atrasos impostos são praticamente coincidentes devido ao facto da segunda fonte terminar a sua transmissão antes da primeira.

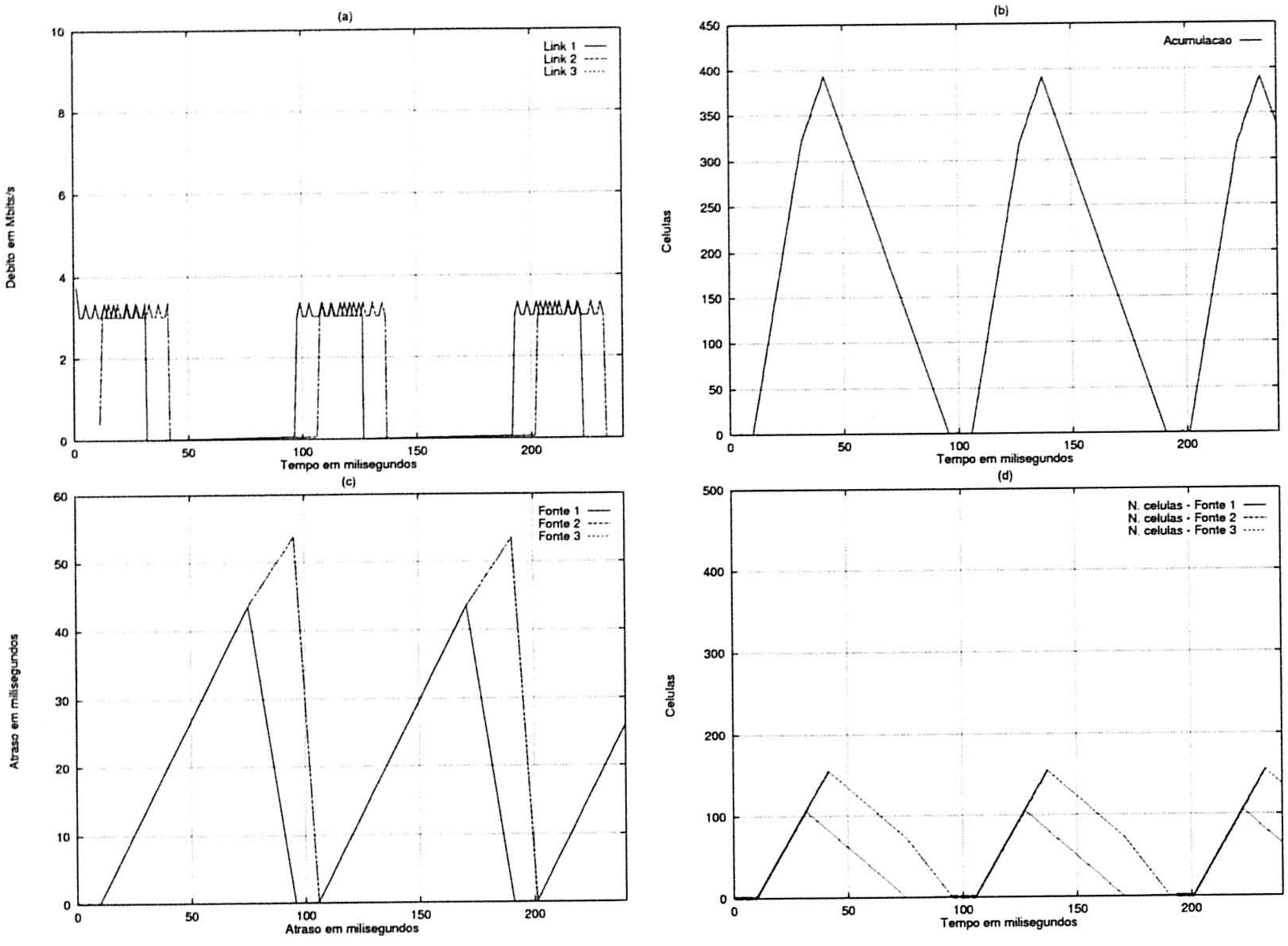

Figura C.3: Resultados obtidos com a disciplina de serviço FIFO no Cenário 3.

Cenário 5

Estes resultados (ver Figura C.5) mostram a ineficiência e a falta de equidade no tratamento das fontes em competição. É suficiente observar o gráfico da ocupação das filas de espera de entrada para verificar que a fonte que inicia mais cedo a sua actividade é beneficiada em relação às outras. O mesmo gráfico permite ainda verificar que em termos de ocupação a última fonte (Fonte 4) é a mais penalizada nos atrasos impostos às células; no entanto é possível verificar que à medida que deixa de existir células para servir na fila de espera da Fonte 1 essa largura de banda é redistribuída pelas outras o que permite que os atrasos aumentem mais lentamente.

Cenário 6

Relativamente aos resultados obtidos com este cenário (ver Figura C.6) não existe nada a acrescentar ao discutido sobre os resultados anteriores, a não ser referir que estes confirmam e acentuam os mesmos problemas.

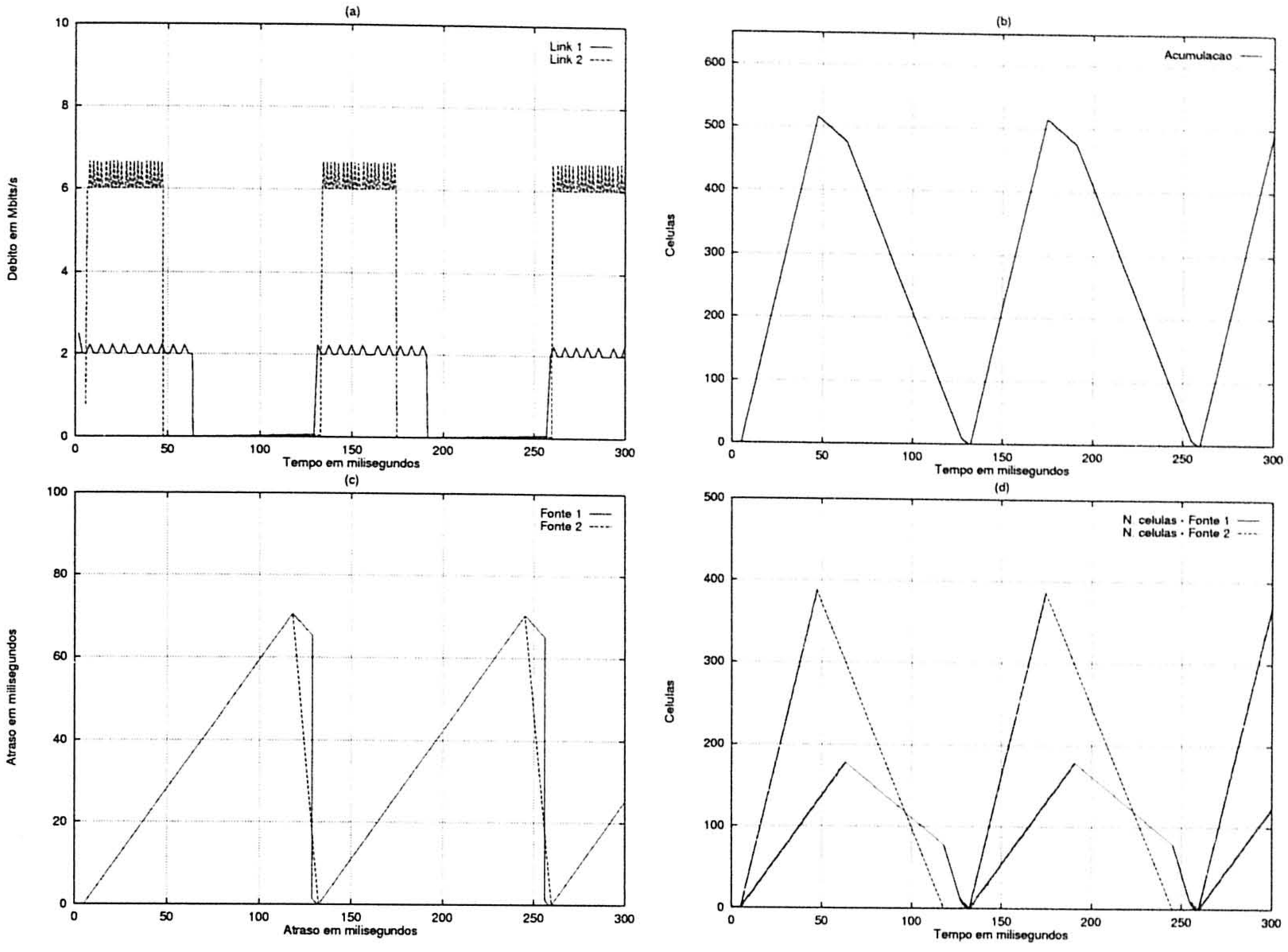

Figura C.4: Resultados obtidos com a disciplina de serviço FIFO no Cenário 4.

C.2.3 Conclusões

Os resultados obtidos com esta disciplina de serviço sobre os cenários definidos mostram que não existe qualquer preocupação na distribuição equitativa da largura de banda quando existem fontes que necessitam de garantias, por exemplo, de um débito mínimo, quanto mais em relação à distribuição dos atrasos.

C.3 Disciplina de serviço WFQ

Antes de apresentar e discutir os resultados obtidos com esta disciplina de serviço é necessário descrever alguns dos pormenores de implementação no simulador.

C.3.1 Detalhes de implementação

A implementação desta disciplina de serviço obrigou a utilizar um conjunto de variáveis específicas de forma a conseguir reproduzir o comportamento téorico. Para implementar a função $F_{i,j}^k$ foi necessário introduzir um campo adicional na estrutura que representa a célula para armazenar o tempo de chegada ($cell.tempoChegada = time()$) que será

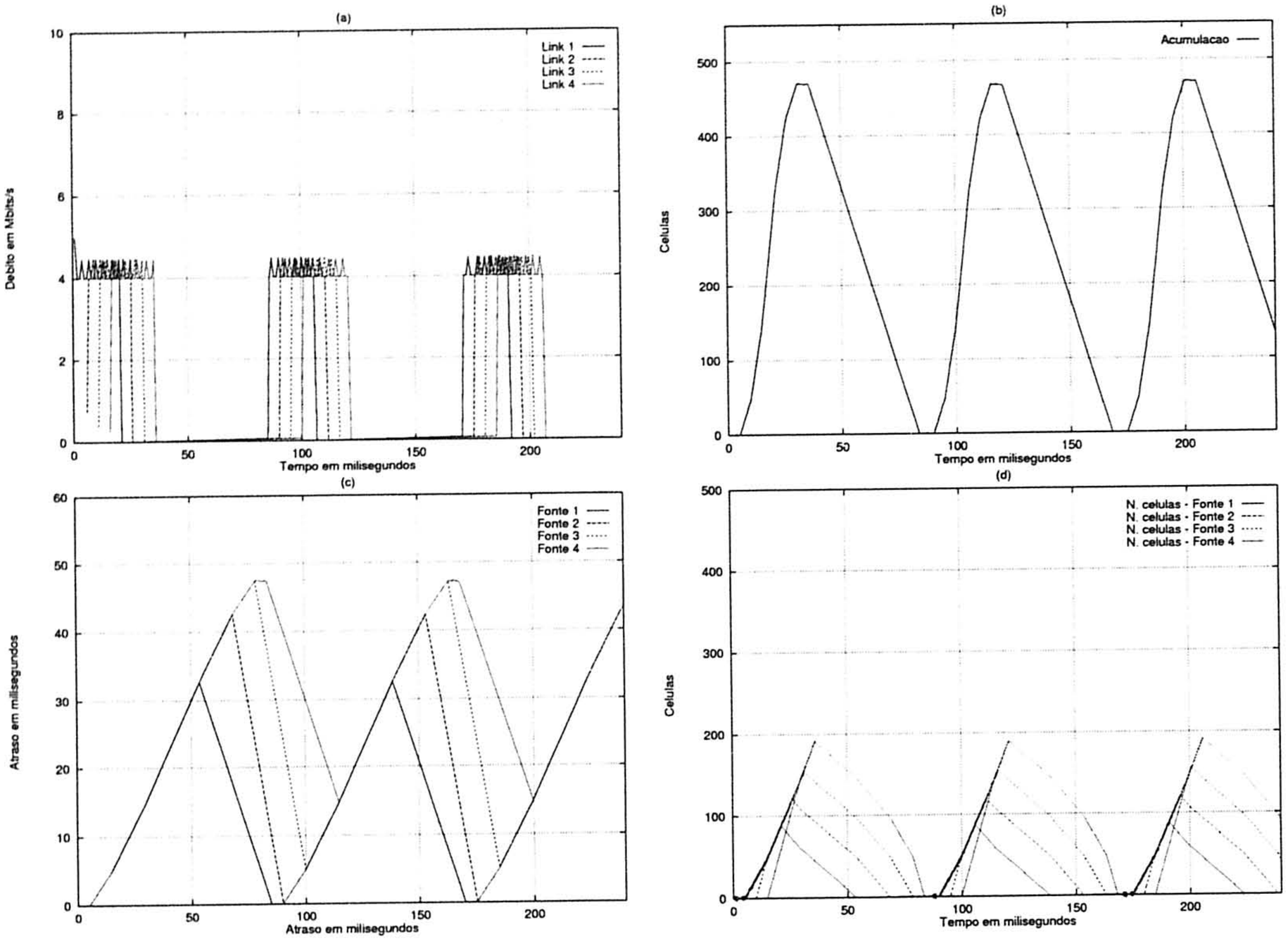

Figura C.5: Resultados obtidos com a disciplina de serviço FIFO no Cenário 5.

utilizado para o cálculo do tempo de partida. De seguida é calculado o tempo de partida da célula ($F_{i,j}^k$) através do valor máximo do tempo de partida da célula anterior e do tempo de chegada da própria, mais o tamanho da célula (L_j^k igual a 424 *bits*) sobre o débito garantido ($\phi_{i,j}$) para esse fluxo. Quanto ao débito garantido e ao tempo de partida da célula anterior foi necessário agregar duas variáveis à estrutura das portas de entrada do ED para armazenar os valores que servirão para o cálculo do tempo de partida da próxima célula que chegar nesse fluxo. Após o cálculo do tempo de partida a célula é inserida ordenadamente pelo menor tempo de partida na fila de saída dos fluxos agregados. Por cada *slot time* é servida a célula que se encontra à cabeça da fila de saída.

C.3.2 Resultados de simulação

Nesta secção serão apresentados os resultados obtidos através da simulação com recurso aos seis cenários (ver secção 5.5) previamente definidos. Estes resultados servirão como base para a discussão relativamente aos méritos e problemas da disciplina de serviço WFQ.

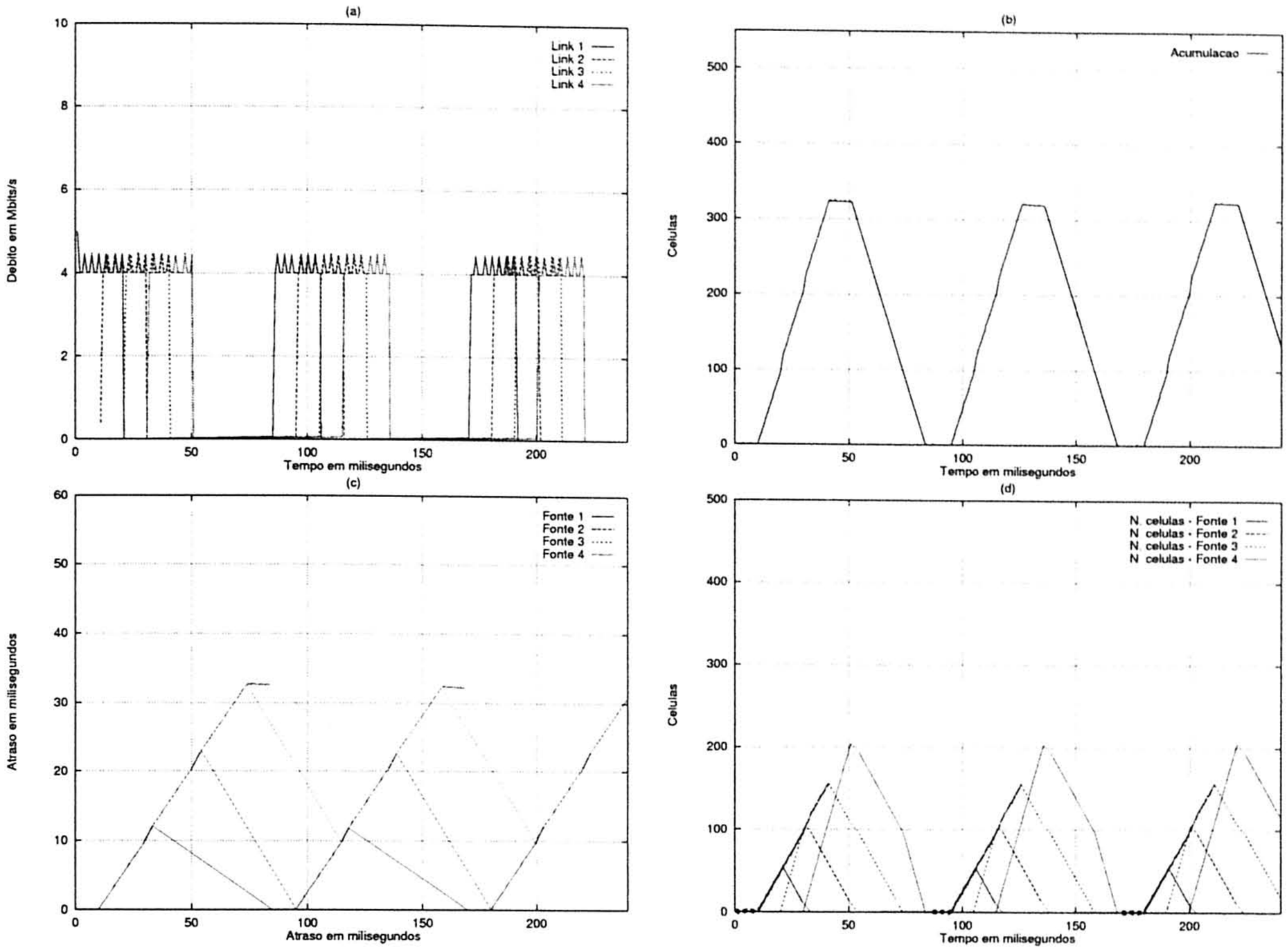

Figura C.6: Resultados obtidos com a disciplina de serviço FIFO no Cenário 6.

Cenário 1

Numa primeira observação (ver Figura C.7) verifica-se que apesar da configuração ser simples, apresenta uma situação de injustiça entre as duas fontes porque a ocupação da fila de entrada da segunda fonte é superior à da primeira, o que se reflecte nos atrasos observados. Estes resultados indiciam que a fonte que inicia a sua actividade mais cedo beneficia em relação à que começa mais tarde. Desta forma é possível concluir que para cenários mais complexos esta situação vai piorar.

Cenário 2

Esta disciplina de serviço realiza a sua função sem criar qualquer tipo de problemas como pode ser observado nos resultados apresentados na Figura C.8, isto é, durante o tempo em que apenas uma fonte estava em actividade, tanto a ocupação da fila de entrada, bem como os atrasos são nulos, mas no instante em que a segunda fonte começa a transmitir o número de células que restam ainda do *burst* da primeira são muito poucas o que levou que a ocupação da segunda fila seja muito maior o que resulta num tempo de

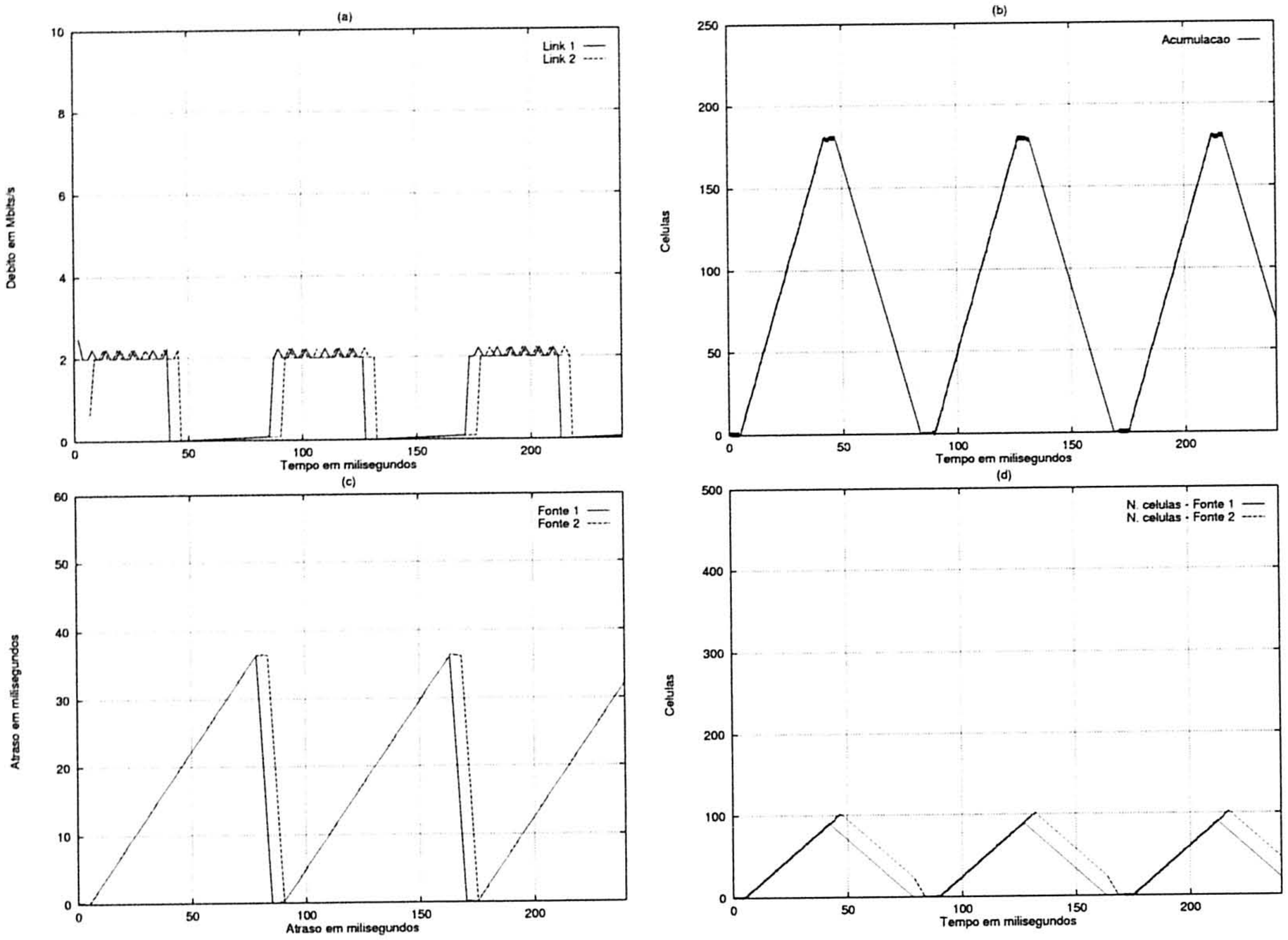

Figura C.7: Resultados obtidos com a disciplina de serviço WFQ no Cenário 1.

atraso elevado para quase todas as células dessa fonte. Desta forma os resultados obtidos com este cenário são iguais aos obtidos com a disiciplina FIFO porque o tamanho do intervalo entre as duas fontes é pequeno para que se possa verificar as vantagens em usar a disciplina WFQ em detrimento da FIFO.

Cenário 3

Globalmente os resultados (ver Figura C.9) são os esperados relativamente à ocupação das filas de entrada e a utilização dos recursos. Quanto à ocupação da filas de espera das fontes pode-se verificar que, por um lado a ocupação observada na fila referente à Fonte 1 apresenta uma ocupação menor do que a fila da Fonte 2 porque inicia a sua actividade mais cedo. Destes resultados é ainda possível perceber que a distribuição da largura de banda e a consequente ocupação das filas de espera de entrada são realizadas de forma equitativa, dado que as fontes que iniciam a sua actividade em simultâneo têm a mesma ocupação e as células sofrem o mesmo atraso. Mais uma vez se percebe que a fonte que inicia a sua actividade mais cedo é beneficiada relativamente às outras que iniciam mais tarde.

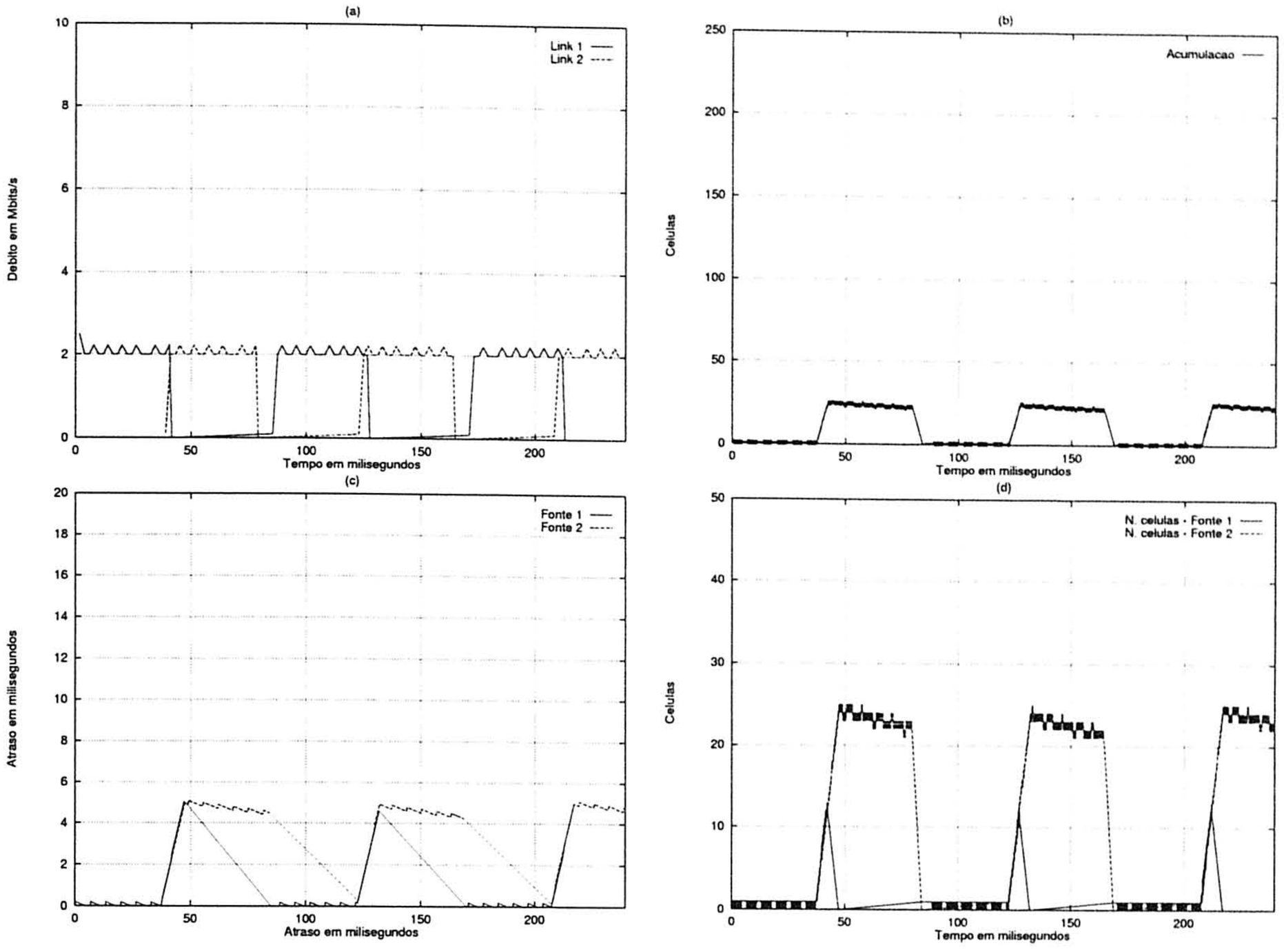

Figura C.8: Resultados obtidos com a disciplina de serviço WFQ no Cenário 2.

Cenário 4

Nos resultados apresentados na Figura C.10 verifica-se a falta de equidade na distribuição dos atrasos porque apesar da Fonte 1 iniciar a sua actividade primeiro não sai beneficiada pois transmite durante mais tempo do que a segunda e assim que a segunda deixa de transmitir, as células que chegam ao ED pertencentes ao fluxo da primeira fonte vão receber uma largura de banda como se estivessem as duas a transmitir, dado que a fila de espera da segunda ainda tem células. Esta situação é um dos problemas desta disciplina de serviço, porque por definição do WFQ os tempo de transmissão das células de um determinado fluxo são calculados com base no tempo de chegada da célula e do tempo de serviço da última célula desse fluxo e do número de filas não vazias. É nesta situação que se encontra o problema, uma vez que o débito é calculado em função do débito garantido para cada fluxo mais o débito disponível distribuído em função das filas de espera com células para serem servidas.

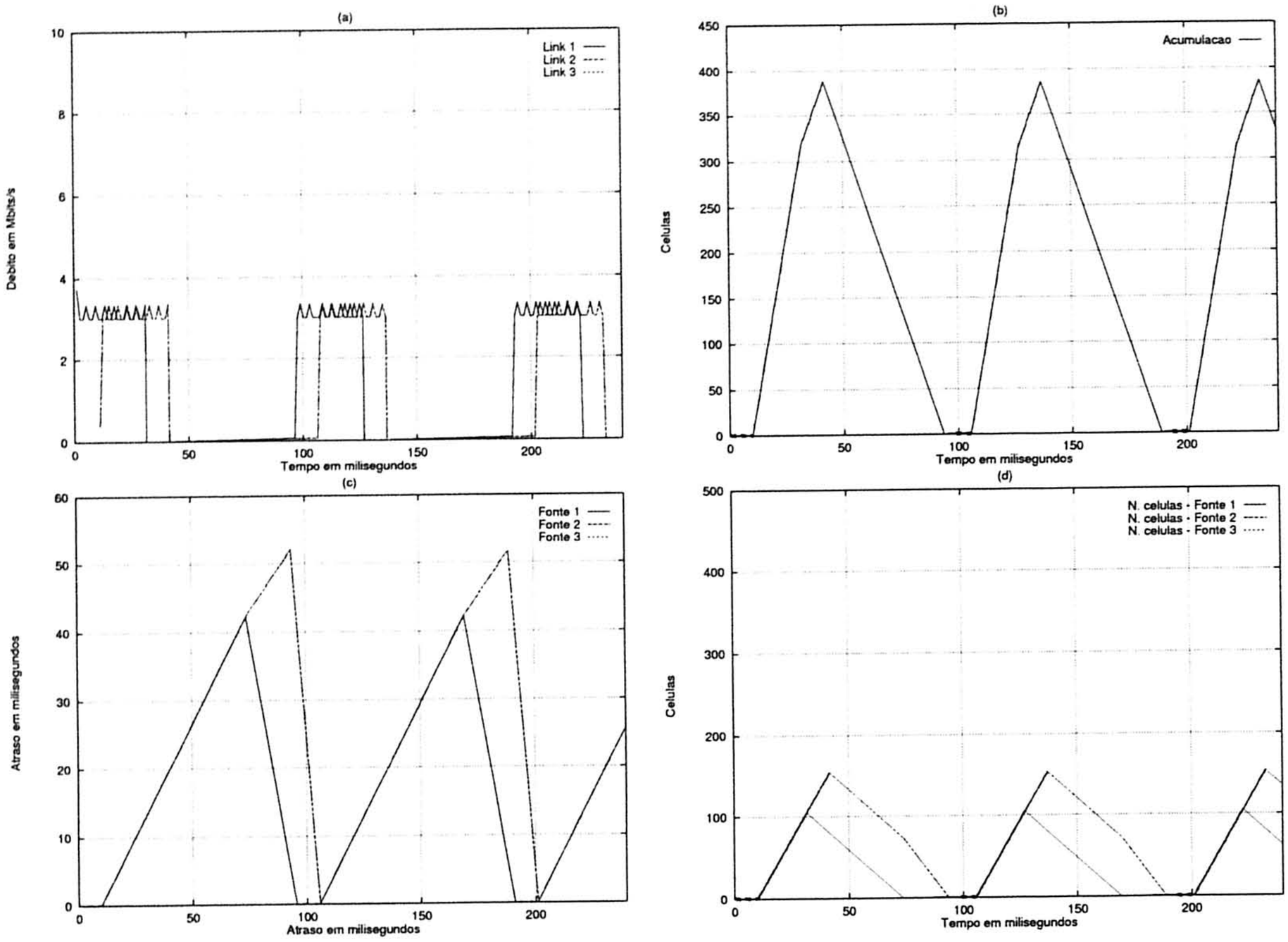

Figura C.9: Resultados obtidos com a disciplina de serviço WFQ no Cenário 3.

Cenário 5

Uma simples observação dos resultados apresentados na Figura C.11, nomeadamente o gráfico da ocupação das filas de espera de entrada, permite verificar que a fonte que inicia mais cedo a sua actividade é beneficiada em relação às outras. Este gráfico permite ainda verificar que em termos de ocupação a última fonte (Fonte 4) é a mais penalizada nos atrasos impostos às células; no entanto pode-se também observar que à medida que deixa de existir células para servir na fila de espera da Fonte 1 essa largura de banda é redistribuída pelas outras.

Estes resultados permitem ainda extrair uma conclusão interessante quanto à distribuição dos atrasos, isto é, a segunda e a terceira fonte são prejudicadas relativamente as outras duas (a primeira e a última a iniciar a actividade). A explicação para este fenómeno é simples porque a primeira fonte é beneficiada inicialmente devido o facto de ter iniciado a sua actividade em primeiro lugar o que lhe permitiu ser servida ao mesmo débito que estava a transmitir e a última fonte vai beneficiar da situação de que ainda está a transmitir o seu *burst* e a fila de espera de entrada da primeira fonte já esta vazia

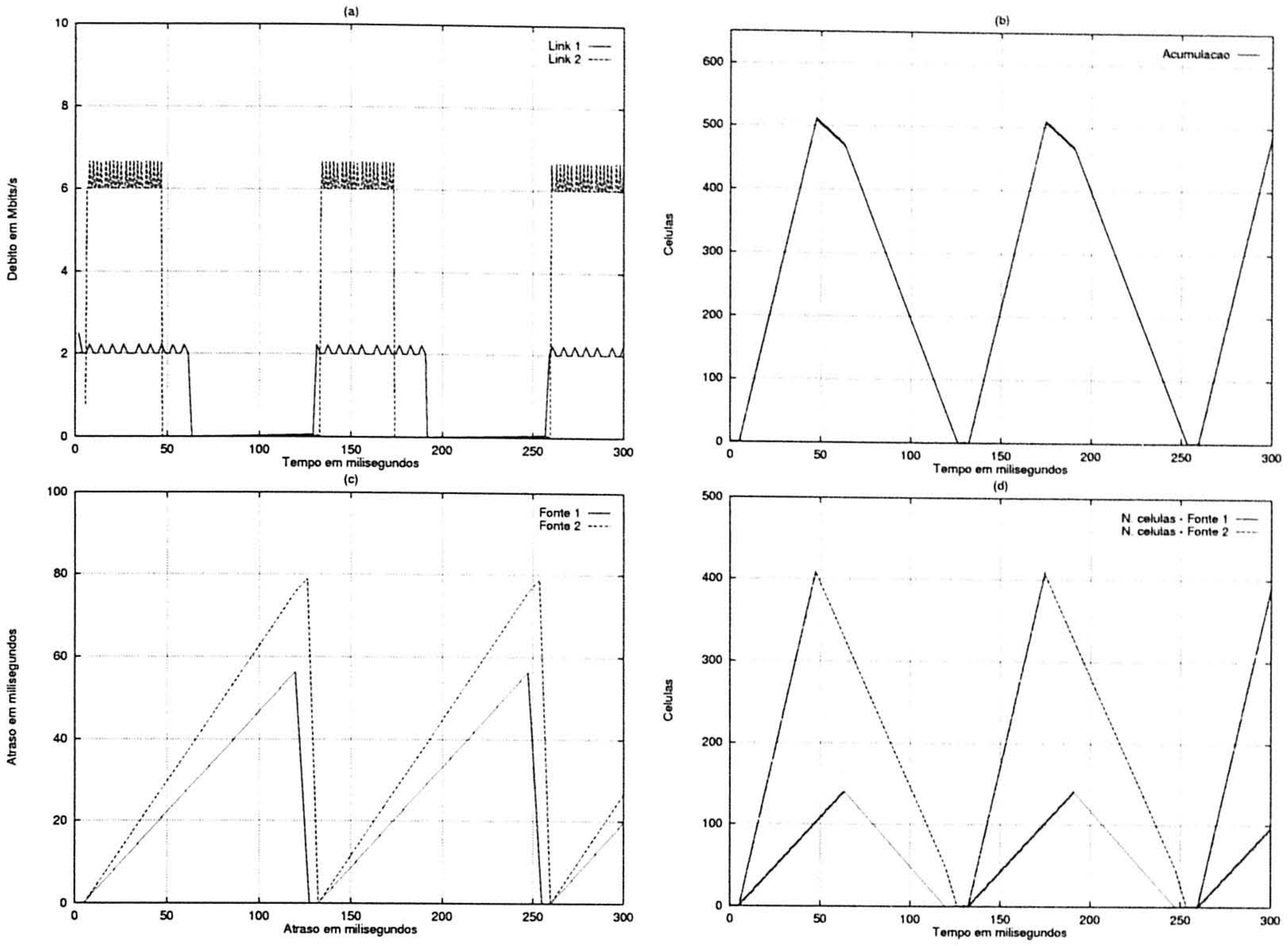

Figura C.10: Resultados obtidos com a disciplina de serviço WFQ no Cenário 4.

e como o cálculo do tempo de transmissão entra em linha de conta com o número de filas de espera de entrada não vazias possibilita a existência de tempos de transmissão diferentes para a quarta fonte.

Cenário 6

Os resultados obtidos com este cenário (ver Figura C.12) não acescentam nada de novo ao discutido sobre os resultados do cenário anterior, a não ser uma confirmação e agravamento dos problemas encontrados.

C.3.3 Conclusões

Sendo esta disciplina de serviço uma referência na literatura porque apresenta excelentes resultados na distribuição da largura de banda a fontes que necessitam de débitos garantidos, para além de obedecer aos limites de atraso impostos, já não consegue os mesmos resultados quando é avaliada relativamente à distribuição dos atrasos e do *backlog* pelas fontes em competição para este tipo de padrão de tráfego.

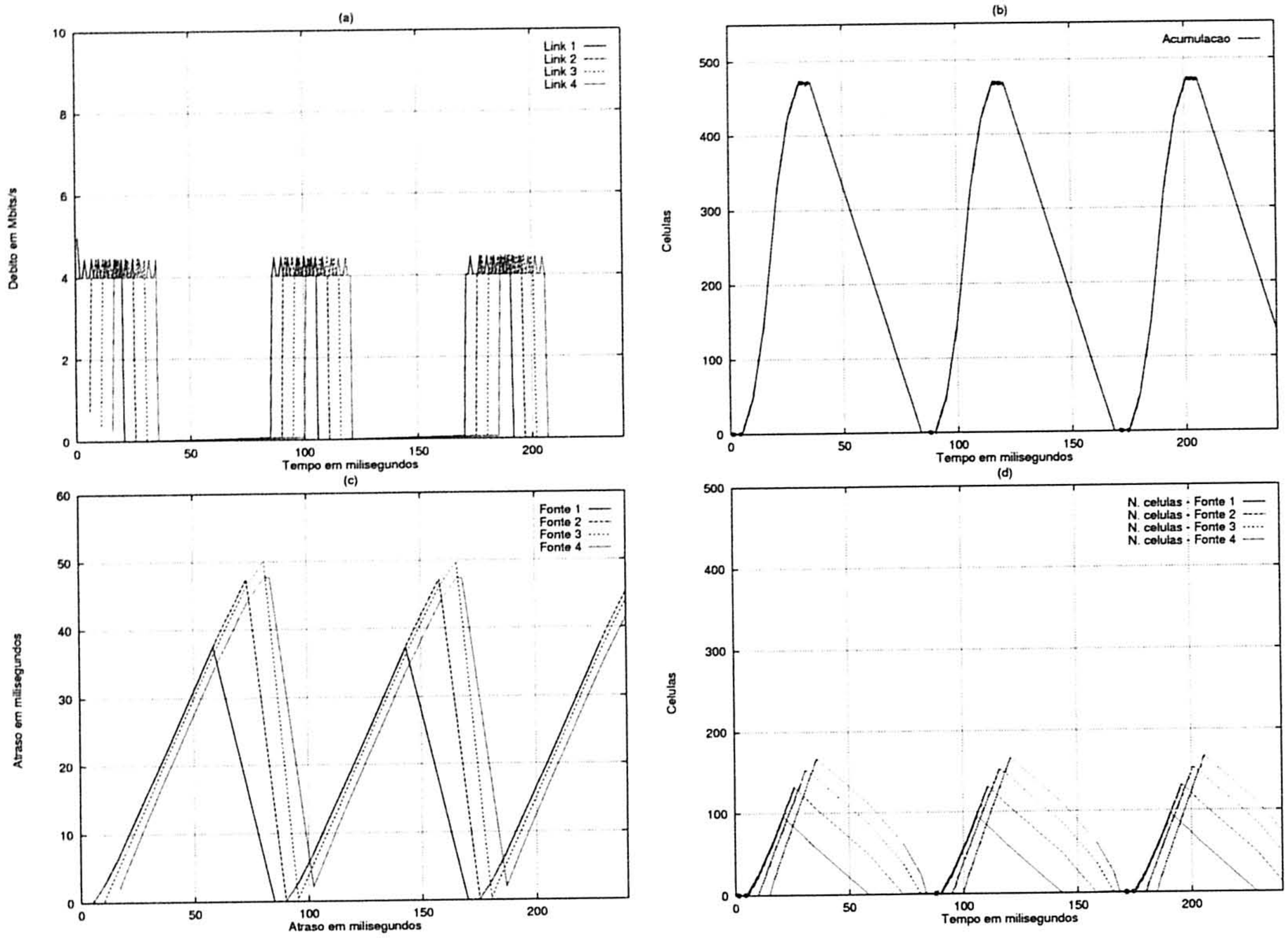

Figura C.11: Resultados obtidos com a disciplina de serviço WFQ no Cenário 5.

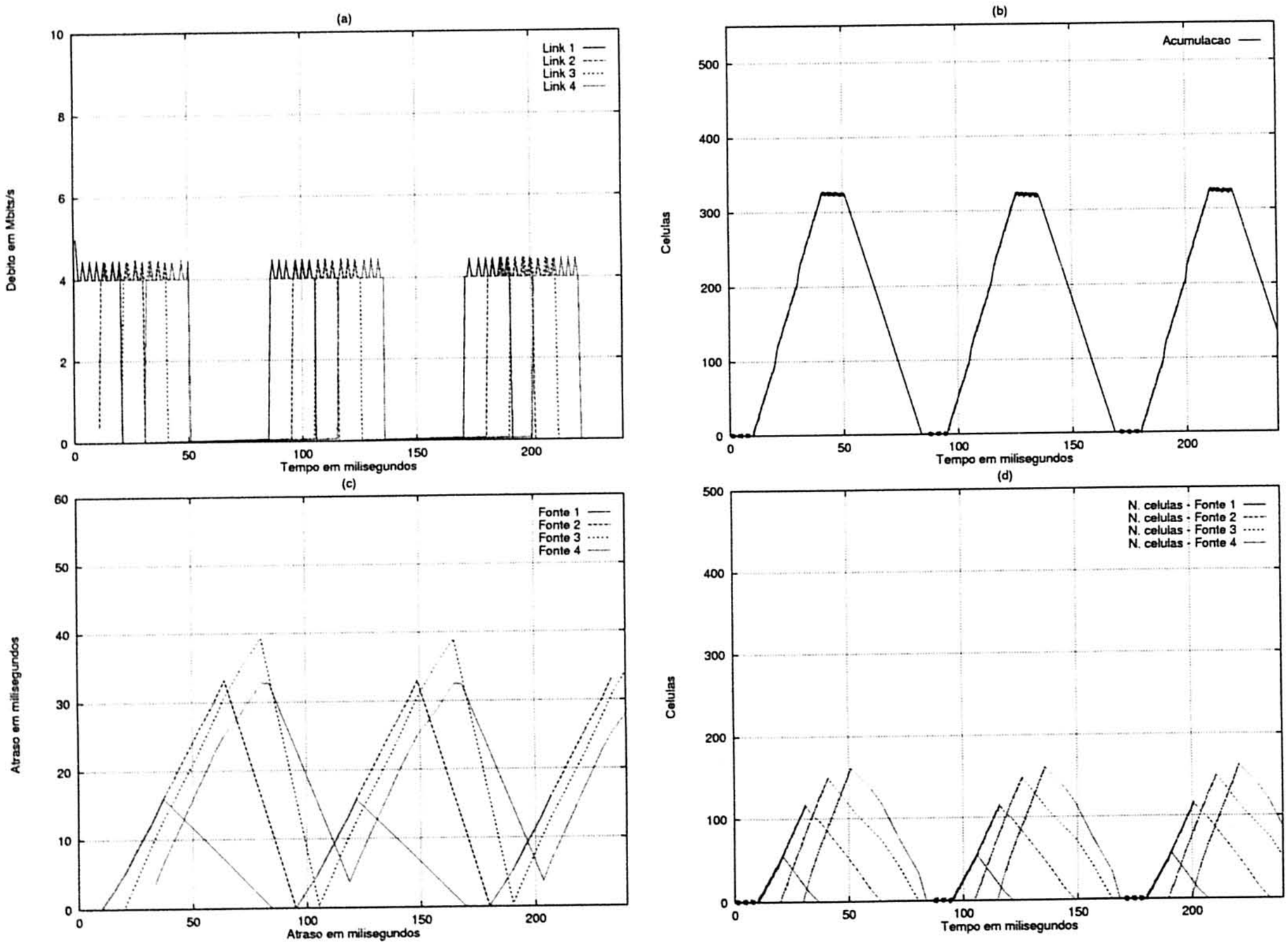

Figura C.12: Resultados obtidos com a disciplina de serviço WFQ no Cenário 6.

Apêndice D

Algoritmos das fontes e dos destinos

O esquema de controlo de fluxo baseado em débitos ficou definitivamente definido pelo grupo de trabalho de gestão de tráfego do ATM Forum. Este grupo fez algumas distinções interessantes relativamente às especificações dos algoritmos dos sistemas finais e do comutador. Por um lado são impostas especificações estritas para o algoritmo dos sistemas finais, por outro lado é deixada em aberto, aos construtores de comutadores, a definição e concepção dos seus algoritmos, desde que respeitem um conjunto de regras mínimas.

D.1 Comportamento dos componentes

O tipo de algoritmo usado nos sistemas finais é apresentado nos documentos do ATM Forum [44, 42] como soluções a implementar e não como meras sugestões.

Antes da apresentação dos algoritmos da fonte e do destino existe a necessidade de definir os parâmetros utilizados nos algoritmos (ver Tabela D.1 [44]).

O comportamento dos sistemas finais veio a sofrer uma sucessiva evolução até à versão final, podendo os vários documentos de trabalho do ATM Forum servir como um bom exemplo para ilustrar esse desenvolvimento. As propostas sucessivamente realizadas, por exemplo, por Andy Barnhart relativamente ao comportamento dos sistemas finais são bastante elucidativas dessa evolução.

O algoritmo da fonte e do destino aprovados pelo ATM Forum são apresentados

Tabela D.1: Parâmetros utilizados nos algoritmos dos componentes

Campo	**Descrição**	**Unidade**
PCR	*Peak Cell Rate*, é o débito que a fonte não pode exceder	Células/Segundo
MCR	*Minimum Cell Rate*, é o débito a que a fonte pode sempre transmitir	Células/Segundo
ICR	*Initial Cell Rate*, débito inicial após um período de repouso	Células/Segundo
RIF, (AIR)	*Rate Increase Factor*, controla o aumento do débito. O *Additive Increase Rate* é AIR = PCR × RIF	RIF é uma potência de 2 Varia entre [1/32768,1] AIR em Células/Segundo
Nrm	Nrm é o número máximo de células de dados que uma fonte pode enviar antes de uma célula RM	Potência de 2 Varia entre [2,256]
Mrm	Mrm controla a distribuição da banda entre uma célula RM enviada, uma célula RM recebida e as células de dados	Valor fixo: 2
RDF	*Rate Decrease Factor*, controla a diminuição do débito	Potência de 2 Varia entre [1/32768,1]
ACR	*Allowed Cell Rate*, débito autorizado da fonte	Células/Segundo
CCR	*Current Cell Rate*, débito corrente da fonte (CCR ≤ ACR)	Células/Segundo
CRM	Limita o número de células RM enviadas na ausência da recepção de uma célula RM	Específico da implementação
ADTF	Tempo permitido entre duas células RM antes de reduzir o débito para ICR	Varia entre [0.01,10.23] seg. com granulalidade de 10 ms
Trm	Limite superior entre células RM	Unidade: milisegundos Trm é 100 × potência de 2 Varia [100 × 2^{-7},100 × 2^{0}]
FRTT	*Fixed Round-Trip Time* é o somatório dos atrasos de propagação da fonte ao destino e voltar	Unidade: 10 μs Varia entre [0,167.77215] seg.
TBE	*Transient Buffer Exposure*, é o número de células negociadas que a rede gostaria de limitar a fonte de transmitir durante os períodos de arranque e antes da primeira célula RM regressar	Unidades: Células Varia entre [0,16.777.215] seg.
CDF	*Cutoff Decease Factor*, controla a diminuição do ACR associado com CRM	0 ou potência de 2 Varia entre [1/64,1]
TCR	*Tagged Cell Rate* limita o débito a que a fonte pode transmitir células RM *out-of-rate*	10 células/Segundo

nas Figuras D.1 e D.2 [44], respectivamente.

O comportamento dos comutadores mereceu uma atenção menor quando com-

1. A fonte começa a transmitir a um débito inicial (ICR) negociado com a rede.
2. O valor do ACR não deve exceder o valor do PCR nem ser menor que MCR. A fonte pode transmitir células a um débito menor ou igual ao valor de ACR.
3. Antes de transmitir a primeira célula RM, as células devem ser enviadas na seguinte ordem:
   (a) A próxima célula a ser transmitida deve ser uma célula RM se e só se, desde a última célula RM transmitida, se
      i. Pelo menos Mrm células foram transmitidas e pelo menos Trm tempo tenha passado, ou
      ii. $Nrm - 1$ células tenham sido transmitidas.
   (b) A próxima célula a ser transmitida deve ser uma célula RM no sentido contrário se a condição (a) não se verificou, se a célula RM no sentido contrário está à espera para ser transmitida, e se
      i. Nenhuma célula RM no sentido contrário tenha sido transmitida desde a última célula RM enviada no sentido do destino, ou
      ii. Não existem células de dados para transmitir.
   (c) A próxima célula a ser transmitida deve ser uma célula de dados se nem a condição (a) nem a condição (b) se verificarem e se existe alguma célula de dados à espera de ser transmitida.
4. Células transmitidas de acordo com os comportamentos da fonte #1, #2 e #3 devem ter $CLP = 0$.
5. Antes de enviar uma célula RM, se $ACR > ICR$ e o tempo T tenha-se esgotado desde a última célula RM transmitida é maior do que ADTF então o ACR deve ser reduzido para ICR.
6. Antes de transmitir uma célula RM e depois de ajustar o ACR de acordo com o #5, se pelo menos CRM células RM tenham sido transmitidas desde a última célula RM recebida com $BN = 0$, então o ACR deve ser reduzido pelo menos de $ACR \times CDF$, a não ser que a redução resulte num débito inferior a MCR, neste caso o ACR deve receber o valor de MCR.
7. Se ACR é reduzido de acordo com #5 e #6, o novo ACR deve ser colocado no campo CCR da célula RM.
8. Quando uma célula RM é recebida com $CI = 1$, o ACR deve ser reduzido pelo $ACR * RDF$, a não ser que a redução provoque uma descida abaixo de MCR, neste caso ACR deve receber o valor de MCR. Se a célula RM recebida tiver $CI = 0$ e $NI = 0$, o valor do ACR deve ser incrementado por $RIF \times PCR$, mas o resultado não deve exceder PCR.
9. Quando uma célula RM é recebida, e o ACR é ajustado de acordo com o comportamento da fonte #8, ACR recebe o valor mínimo entre o ACR calculado em #8 e o valor do campo ER, mas nunca abaixo de MCR.
10. Quando gera uma célula RM, a fonte deve atribuir os valores aos vários campos da célula.
11. Células RM transmitidas pela fonte podem ser enviadas *out-of-rate* (sem estar em conformidade com o valor de ACR). Célula RM transmitidas *out-of-rate* não devem ser transmitidas a um débito superior a TCR.
12. A fonte deve inicializar o campo EFCI de todas as células de dados que vai transmitir.
13. A fonte pode implementar uma política de *use-it-or-lose-it* para reduzir o seu ACR para um valor que se aproxime do actual débito de transmissão.

Figura D.1: Algoritmo da fonte.

1. Quando uma célula de dados é recebida, o seu EFCI é guardado como o estado EFCI da conexão.
2. Na recepção de uma célula RM, o destino deve devolver a célula para a origem. O campo DIR da célula deve ser alterado de *"forward"* para *"backward"*, o campo BN deve ser atribuido a zero e CCR, ER, CI e NI devem ser mantidos inalteraveis excepto se:
   (a) Se o *bit* EFCI está atribuido, então o destino deve colocar $CI = 1$ na célula RM e o *bit* EFCI deve ser inicializado.
   (b) Se o destino tiver congestionamento interno deve reduzir o valor do campo ER para o débito que ele pode suportar e/ou atribuir $CI = 1$ ou $NI = 1$.
3. Se a célula RM é recebida pelo destino enquanto outra célula RM está a ser escalonada (na mesma conexão) para transmissão:
   (a) É recomendado que o conteúdo da célula mais antiga seja substituido pelo conteúdo da última célula;
   (b) É recomendado que a célula mais antiga deve ser enviada *out-of-rate*; alternativamente a célula mais antiga pode ser eliminada;
   (c) É requerido que a nova célula seja escalonada para transmissão.
4. Para além das alternativas de #3, o conteúdo da célula RM antiga não deve ser transmitida depois do conteúdo da nova o ser.
5. O destino pode gerar células RM sem ter recebido nenhuma célula RM. O débito destas células RM deve ser limitado a 10 células/segundo por conexão. Quando o destino gera uma célula RM ele deve atribuir $CI = 1$, $NI = 1$, $BN = 1$ e $DIR =''$ *backward*$''$. O destino deve atribuir os valores dos vários campos da célula RM.
6. Quando uma célula RM com $CLP = 1$ é devolvida ela pode ser transmitida *in-rate* ou *out-of-rate*.

Figura D.2: Algoritmo do destino.

parados com os sistemas finais. Esta afirmação é justificada através das Versões 4.0 [44] e 4.1 [42] do ATM Forum, onde é fornecida uma especificação bem definida para os sistemas finais e são deixadas apenas algumas pistas para os algoritmos dos comutadores.

O ATM Forum classifica os algoritmos dos comutadores com base em três características:

- Cálculo da distribuição equitativa da largura de banda disponível pelos VCs activos;
- Determinação da carga da rede, em especial a percentagem da ocupação das filas de espera no comutador;
- Cálculo explícito do débito, ao qual as fontes devem transmitir.

Com base nestas três características os algoritmos propostos pelos seus membros estão divididos em três categorias diferentes:

- EPRCA;
- Esquema para a Antecipação de Congestionamento (*Congestion Avoidance Scheme*), caracterizado por um factor que indica a carga no comutador:
  - Indicação Explícita do Débito para Antecipação de Congestionamento (*Explicit Rate Indication for Congestion Avoidance*, ERICA);
  - Antecipação de Congestionamento Através de Controlo Proporcional (*Congestion Avoidance using Proportional Control*, CAPC);
- *ER based on bandwidth Demand Estimate Algorithm.*

O primeiro tipo de algoritmo faz uso do campo CI da célula RM para indicar à fonte se ela deve diminir ou aumentar o seu débito. Além da indicação do congestionamento o comutador fornece o débito explícito que a fonte não pode ultrapassar. O segundo e o terceiro tipo de algoritmo caracterizam-se pelo cálculo explícito do débito nos comutadores e pela prevenção do congestionamento através da taxa de ocupação das filas de espera. Nestes algoritmos os campos NI e ER são utilizados para controlar o comportamento do sistema e o débito de transmissão das fontes. Os algoritmos que se enquadram nesta classificação são: algoritmo do NIST e o algoritmo de Barnhart.

BIBLIOTECA F.E. PORTO

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO
BIBLIOTECA