Modelling and Planning the Migration for Next Generation Networks

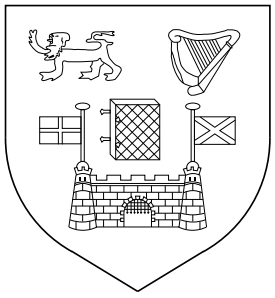
by

Mathieu Robin

A thesis submitted to
the University of Dublin
for the degree of

Master of Science in Computer Science

Department of Computer Science,
University of Dublin, Trinity College



September, 2002

Declaration

This thesis has not been submitted as an exercise for a degree at any other University. Except where otherwise stated, the work described herein has been carried out by the author alone. This thesis may be borrowed or copied upon request with the permission of the Librarian, University of Dublin, Trinity College. The copyright belongs jointly to the University of Dublin and Mathieu Robin.

Signature of Author .....

Mathieu Robin

16 September, 2002

Acknowledgements

I would like to thank my supervisor, Meriel Huggard, for her enthusiasm and guidance throughout the course of this project. I would also like to thank the people at Eircom involved in this project, David Blair and Lorcan Dillon-Kelly. Finally I would like to thank my friends and classmates for their continuous support throughout the year.

Abstract

The ever increasing volume of data traffic on telecommunications networks has led to a fundamental shift in network operator requirements. Moreover, the continuous evolution of the industry, with developments such as mobile telephony, has created a need for interoperability and advanced service management.
These changes in network usage have given rise to major infrastructural change: telecommunication networks are moving from switched circuit networks, designed to handle telephony service, to multi-service digital technologies. This evolution is referred as the migration to Next Generation Networks.

The development of a computer-based tool for modelling and planning this migration is outlined. This may be used to study the main characteristics of the new network infrastructure. The tool focuses on network dimensioning and quality of service: Algorithms have been created to study how migration affects capacity requirements and a network simulator has been developed to study and analyse quality of service.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis describes the work carried out to design and implement an improved version of a computer-based tool to model the migration to Next Generation Networks (NGN).

Telecommunication network operators use the term *migration to Next Generation Networks* to refer to the process of moving from a network infrastructure designed to handle voice communication to a multi-service digital one.

1.1 Context

The use of the telecommunication networks to carry data and real-time services has changed the requirements in terms of infrastructure and technology. Currently, services tend to be provided on dedicated networks. It is envisaged that these will be replaced by NGNs which provide integrated multi-service networks.
Today, the telecom operators are engaged in modifications and updates of their transport network. Even if the technologies involved are well established, for example IP and ATM, the path of migration is still unclear.
This migration has started in the mid 1990s, but most operators still have to make decisions regarding the development of their NGNs and the integration of the legacy

infrastructure. As stated in [Don01], the motivations behind the migration are:

- The high growth of the Internet, which implies an increase of a world wide IP traffic,
- The growth of mobile telephony, and the change to a multi-operator competitive environment,
- The increasing complexity of network and service management.

However, a lot of issues arise regarding the deployment of the NGNs:

- *Cost*: The migration has to be cost effective, bearing in mind that the revenue brought in by classic telephony is decreasing.
- *Quality of Service*: The customers expect perfect quality of service (QoS) for voice traffic, and such QoS is not easily achievable on cell/packet switching networks.
- *Planning*: Forecasting of demand and network planning are harder for data traffic; the models involved are more complex than the ones used for telephony.
- *Underlying technology*: For packet-based, multi-service networks, two technologies exist: IP and ATM. However, it is difficult to choose between these; both are supported by different players of the telecommunications community.

The use of planning tools to assist telecommunication operators in their investments through the migration steps is needed.
We will detail NGNs and the migration to these networks in chapter 2.

1.2 Project Goal

The project statement is to provide easy and fast ways to forecast the advantages of the migration to NGNs, and to provide a quick and fast overview of the network performances given the technology used.

As this work is based on a project funded by Eircom ([Hov01]), the collaboration with this telecom operator has been used: Eircom has provided information regarding traffic model, a realistic network infrastructure, and some general consideration regarding the migration to NGNs.

1.2.1 Previous Work

The main goal of this project is the validation and the improvement of a computer-based tool developed by Eirik Hovland in 2001 ([Hov01]).
This tool has two main features:

- *Traffic growth planning*: Using the Kruithoff algorithm, the tool is able to estimate traffic needs for the next three years.

- *Bandwidth requirement estimation*: Given the amount of data transfered during one peak hour, the tool computes an approximation of the bandwidth required to carry such data efficiently. The source model is taken from the ATM specification, and sources are supposed to some ATM traffic class. The underlying network can be PSTN (Public Switched Telephone Network) or ATM.

The most useful feature of this tool has been the bandwidth estimation function. It allows an easy comparison between the PSTN network and the ATM network, by estimating the bandwidth gained using ATM instead of PSTN-based networks. Various network scenarios and traffic mixes may be studied using the planning tool.

1.2.2 Validation of Previous Work

The traffic growth planning feature has not been implemented in the new version, given the difficulty of traffic growth forecasting in NGNs. Efforts have been focused on improving the second feature, and adding valuable ones to the tool.
During this first phase, efforts have been focused on improving the scalability and the quality of the metrics provided.

1.2.3 Improvement: Network Simulation

After a first step, during which the algorithms used in Hovland's project has been implemented and validated, it has been decided to improve the tool by building a small network simulator on top of the existing network model, to allow dynamic studies and not only static bandwidth requirements. The network simulator brings data such as the delay or the loss ratio in the evaluation of a system.

1.3 Thesis Outline

The next two chapters describe the background research which has been used to design the second step of the project. Detailed information about the Next Generation Networks and the current use in the industry are in chapter 2.
The study of network simulation and the implementation of a simple network simulator represent a significant part of this project. Chapter 3 is a study of the main algorithms used in network simulation, issues which are rising then writing a network simulator, and a short review of the simulators currently available on the market.
The general design of the two versions of the planning tool is described in chapter 4. Details about the implementation of [Hov01] is given in chapter 5, a review of the key algorithms of the actual implementation of the network simulator is given in chapter 6. Various tests have been carried out, the method used and the results can be found in chapter 7.
Finally, chapter 8 concludes this thesis, with a detailed analysis of the work carried out and further work propositions.

Chapter 2

Migration to Next Generation Networks

2.1 The Next Generation Networks

For the last 15 years, the telecommunication world has to face new challenges. The infrastructure and technologies designed to carry only telephony services need to evolve to cope with the expanding of multimedia and data services. According to [Don01], "*the most striking changes have been in the growth of both the Internet and mobile telephony*". The new multi-services network has to be build on top of the legacy equipment of the PSTN, and it is essential that this transition is both smooth and cost effective. This transformation is referred as the migration to Next Generation Networks, or NGN. In simple terms this can be described as a transition from circuit switching to cell switching or packet forwarding.

As stated in [Eri01b] and many other papers, the two technologies of choice for NGNs are ATM and IP. This chapter will focus on the main advantages and drawbacks of these technologies.

The main challenge is to provide the customer with the same features he had on PSTN networks: as a total switched connection protocol it provides a perfect connection between two users. The advances in technology and customers demand are now bringing

about the rise of broadband multi-service networks ([Eri01a]). Bennett ([Ben01]) outlines the importance of reliability on voice over packet connections. Another challenge facing network operators is to adapt the billing strategies for these technologies. Most of the following considerations about ATM and IP have been gathered from various sources, especially [Dan02] and [PD00].

2.2 IP - Internet Protocol

The main characteristics of IP are that it is:

- the underlying technology of the Internet and of almost every LANs,
- highly flexible, and easy to manage,
- designed to scale well and to be fault tolerant,
- the technology of choice for future multimedia services,
- able to provide QoS and traffic engineering (although these are recent additions to the protocol and are not yet widely applied).

2.2.1 IP Telephony

One of the major fact in favour of the use of IP for NGNs is the growing use of this technology to carry telephone traffic.
The main benefits of IP telephony are that:

- Customers can take advantage of flat Internet rating to save money on long distance calls. While classical telephony rates depends of distance and can be significantly high, the cost of an Internet connection is constant. However, the prices for both technologies are likely to converge as IP telephony gains market shares.

- The telephony service could be integrated within the computer environment, allowing for the management of phone calls and phone books from sources such as customer relationship management, sales force automation, supply chain management, time accounting, etc. (Source: [Gro01]). The integration and extension of software solutions should reduce the investment in terms of time and money.

In this section, a short overview of the two main protocols of Voice over IP is given. These protocols are H.323 and SIP (Session Initiative Protocol).

H.323

H.323 is an ITU-T (International Telecommunication Union) specification for multimedia conferencing over IP. It was designed to address a concern for the requirements of multimedia communication over IP networks, including audio, video, and data conferencing. The data is encoded in a compact binary format that is suitable for both narrowband and broadband connections.
H.323 is designed as an unified system, which allows interoperability with legacy telephony systems, along with a level of robustness and reliability that is comparable to the one of current PSTN technology. Actually, H.323 includes many other standards, for signalling, real-time voice transports, codecs. It is a peer-to-peer protocol, while other protocols, such as MGCP[1], imply a central control.

SIP

SIP is an IETF (Internet Engineering Task Force) standard for peer-to-peer multimedia sessions and IP telephony. It is designed to setup a session between two points, using the Internet architecture. The guarantees of QoS and connectivity are quite low; SIP has no support for multimedia conferencing and interoperability is not well supported.

[1]The Media Gateway Control Protocol was designed to simplify the design and cost of IP telephony devices by moving the complexity to the network core.

SIP messages are encoded in ASCII text format: the messages are large and less suitable for networks where bandwidth, delay, and/or processing are of concern.

Comparison

H.323 is an adaption of existing technology to the Internet, and designed to inter operate well with the Plain Old Telephony System (POTS). On the other hand, SIP has less features of the old telephony service, but has been designed to be integrated among the other Internet applications, allowing telephony from the computer desktop. As IP does not provide control over intermediate routers involved in a connection, the only way to ensure good QoS is the use of dynamic routing ([Net99]). Both of the technologies discussed use dynamic routing. Another common feature is that these technologies are peer-to-peer.
The differences between the two protocols are due to their respective promoters:

- H.323 has been designed by ITU-T, is promoted by the telecom industry, and deals with integration with the telecommunication infrastructure;
- SIP is an IETF standard, and thus can be seen as an Internet-based application.

2.3 ATM - Asynchronous Transfer Mode

ATM is a connection-oriented technology, based on packet switching. It rose to prominence in the 1980s and in the early 1990s, because it was the technology of choice of telecom industry, and at this time had a lot of interesting features for LAN networking.
The main advantage of ATM compared to IP is the built-in QoS features. While these have been added to IP or to over-IP protocols, ATM was designed to handle them. The connection oriented features of ATM, along with the classification of traffic and Connection Admission Control (CAC) allow for powerful management of QoS:

- Connection Oriented: a virtual circuit is created along all the ATM switches during an ATM transfer; the bandwidth is allocated.

- Classification of traffic: ATM maintains a number of traffic classifications, which characterise the most common traffic types. Sources on ATM networks have to conform to one of these traffic types, so that QoS control of the sources as well as the switches is possible.

- CAC mechanisms ([PE96]) used with the appropriate class of traffic enable the application of QoS management policy directly on the ATM switch.

2.3.1 ATM Traffic Types

The ATM traffic types classification is used to determine which cells should be discarded, as well as how flows are interacting with each other. At connection setup, QoS parameters are negotiated between the application which is sending data and the CAC part of the ATM switch. We now consider some of the main traffic types.

CBR - Constant Bit Rate

CBR is used to model traffic generated by applications which require a guaranteed constant level of bandwidth, such as uncompressed voice traffic. Cells exceeding the negotiated bit rate are discarded.

VBR - Variable Bit Rate

VBR is divided into two subclasses: VBR-rt and VBR-nrt, respectively *real time* and *non real time*. VBR traffic must conform to some specific characteristics. Bandwidth allocation is negotiated and guaranteed according to these specifications. The traffic is modelled as a bursty source: two bitrates are associated with, SCR and PCR. SCR represents the normal bitrate, while PCR represents the maximum rate during a burst. Another parameter is the MBS, Maximum Burst Size. The source has to conform to this model and non-conforming cells may be discarded.

ABR - Available Bit Rate

Sources complying to ABR must change their bitrate according to the information sent by the switches. These switches can send three different types of command: increase, decrease, or maintain bitrate. This behaviour can be approximately compared to TCP/IP traffic, which can adjust its rate. However, for TCP the source does the changes by itself.

UBR - Unspecified Bit Rate

UBR is the less restrictive ATM traffic class. Thus, it is given a low priority on switches. Like ABR, UBR is designed to represent data transfer which is insensitive to delay or jitter.

2.3.2 Drawbacks

ATM has some drawbacks. The cell size is fixed with a relatively important overhead (5 bytes out of 53). This choice has been made as a compromise between telecom and network industries. The need for a small fixed cell size is a requirement for carrying real-time traffic.
Some mechanisms, called ATM Adaptation Layer (AAL) are designed to allow transport of various protocol over ATM. The idea is to provide a layer between the upper protocol and ATM, to disassemble and reassemble the message. This AAL adds up to 4 extra bytes of overhead with a total of 17% overhead on the payload.

2.4 Quality of Service in IP networks

The matter of quality of service within IP is made especially difficult as the source and the destination of a data transfer have no control on the intermediate routers on the data path.
According to Peuhkuri in [Peu99], there are currently two main efforts to improve

QoS support in IP: the Integrated Services and Differentiated Services, usually named intserv and diffserv.

2.4.1 Integrated Services

To increase the support of QoS in the Internet, an IETF Internet Integrated Services group was formed. It defined a framework, independent from signal protocols and implementation, that enables resource reservation and performance guarantees in IP networks.

Service Classes

There are currently two service classes, *Guaranteed Quality of Service* and *Controlled-load Network Element Service*.

- **Guaranteed Quality of Service**: has been designed for applications which require some minimum bandwidth and maximum delay. The traffic has to conform to a given fluid specification. The router must transmit the conforming packets according to receiver specification. Non-conforming packets are considered as best-effort packet, and have no advantage over regular best-effort packets.
- **Controlled-load Network Element Service**: this service is also based on a fluid specification that flows must conform to. It provides conforming flows a QoS, that approximates the one it would get in an unloaded network. Capacity admission control is used, and non-conforming packets are treated as regular traffic.

RSVP - Resource Reservation Setup Protocol

RSVP is an Internet protocol being developed to enable the Internet to support QoS. Using RSVP, an application will be able to reserve resources along a route from source to destination. RSVP reservation is initiated by the sender: it sends a `path` message which records the intermediate routers on the path, the recipient then replies

with a `resv` message, which actually reserves resource on every router on the way back, along the same path.
The main issue in RSVP is the control the network operators can have on it. Without control, all users will want to reserve bandwidth. As no billing system has been defined, the deployment of RSVP is difficult.

Intserv and ATM

With the use of RSVP, route reservation makes IP more similar to ATM. However, there is no permanent connection, as RSVP has to do periodic updates to maintain the reservation.
The services classes defined above can be mapped to ATM class quite easily. The mapping is shown on table 2.1. In this table, the last line *'Best Effort'* corresponds to a regular unmanaged IP data transfer.

Intserv	**ATM**
Guaranteed Service	CBR or rt-VBR
Controlled Load	nrt-VBR or ABR
Best Effort	UBR or ABR

Table 2.1: Intserv/ATM Service Class mapping

2.4.2 Differentiated Services

In Intserv, the routers have to maintain information about all the flows which are reserving resources at this node. If this method allows the network to provide well-defined QoS, it does not scale well.
In Diffserv, the packets are classified at the edge of the network. They are given a Differentiated Service (DS) codepoint. This codepoint is then used in the core of the network to forward or drop the packets on a per-hop basis.
Diffserv scales well, as it is independent of the application, is working even if some

parts of the path are not diffserv-enabled, can inter-operate with other QoS technologies, and does not need to keep information about the flows.

These services are representatives of the efforts currently made to enable QoS support in IP, a protocol which was only designed to handle data transfers on unreliable networks.

2.5 IP vs. ATM

About 8 years ago, ATM was seen as the future underlying technology for a lot of different networks. However, the improvements in LANs domain made by Ethernet technologies, in speed and in price, made ATM far less attractive for use in this environment. The development of switched Ethernet has also been a major improvement to Ethernet technology. With the recent innovation of Gigabit Ethernet, and soon 10 Gigabit Ethernet, ATM is even losing his advantages for use in MANs (Metropolitan Area Networks). Nowadays, it is mostly used by telecom operators in network backbones or to extend local access.
However, ATM built-in traffic classes allow for an easy characterisation of the QoS abilities offered to a customer by a telecom operator. Efforts are being made ([CGK02]) to improve this aspect of IP.

The importance of IP, with the use of the Internet, VPN, and VoIP technologies is rising. The problem is that ATM is not designed to carry IP traffic very well. The overhead added is about 15% of the data transmitted. The growing importance of IP-based applications in the overall traffic is certainly another drawback for ATM-based solutions. In [GW96], Gurski and Williamson report on the poor performance of TCP over ATM, due to the poor interaction of two different designs, aimed at solving different problems.

2.6 NGN Planning

This section explains the example of Eircom migration plan ([Don01]). It is used to illustrate the migration towards NGNs. This is just one possible example; almost all telecommunication operators will have to create similar plans.

2.6.1 Current Network

The current network infrastructure used by Eircom is made of a telephony network and an ATM network. This one represents the first step in Eircom migration to a future multi-service network, carrying voice, IP traffic, legacy data, and eventually leased lines. A smaller, separate ATM network has been deployed to support ADSL deployment. The PSTN traffic is converted to ATM traffic by Media Gateways.

2.6.2 New Network Architecture

Today mobile telephony, PSTN/ISDN, Data/IP networks, Cable TV, and soon wireless local loop, all have their own single-service networks.
The new network architecture will be made up of:

- a multi-service connectivity core,
- services and content providers running as applications on dedicated servers. This is called application layer, and the services can be implemented by the operator or by third-party service providers.
- multi-technology access systems connected to edge gateways, with control for service access and connections across the core.

A detail view of the core of the NGN:

- The transmission is all optical, based on Dense Wavelength Division Multiplexing (DWDM) and other optical components.

- The transmission layer is based on IP routers, ATM switches or combinations of both.

- The core is interconnected to other services and other operators through media gateways (ATM edge switches or IP edge routers).

Features of the core include: interoperability, scalability (capacity and geographically), reliability (Quality of Service or Guarantee of Service), adherence to standards (to ensure good inter-working).

2.6.3 IP vs. ATM, Eircom Point of View

Currently, the preference of Eircom is for an ATM-based network. This technology is favoured over IP, because of their existing ATM network, of the QoS built-in guarantees of ATM, and its ability to carry high-quality voice traffic. This situation represents well the point of view in the industry: the choice between ATM and IP is driven by the existing network and the needs of the operators, no technology is intrinsically better than the other for everything.
However, an overlay IP business network will be developed to complement the ATM core. The core will be *hybrid*, between the two extremes, *'all-ATM'* network or *'IP-only'* network, supported by a new generation of ATM/IP integrated switches.

2.6.4 Migration Path

The evolution chosen by Eircom is a migration to a hybrid platform. This evolution will be done in two phases:

1. Installation of IP routers in the core network. At this stage, different services will be offered to customers: full QoS guarantee relying on the ATM network, and lower quality, but cheaper, services running on top of the IP network.

2. Evolution of the core nodes to ATM/IP hybrids. Much of the development of hybrid node is built with the use of Multi-Protocol Label Switching (MPLS) for

IP. This technology adds labels to IP headers to simplify routing. It provides a good method to carry IP traffic over ATM and enable some sort of QoS support in IP. Thus, the hybrid ATM/IP networks are referred as *'MPLS-based'*

The migration path for telephony is still not clear. However, the general strategy is to convert some PSTN switches to core gateways; the importance and the depth (in the network hierarchy) of these changes is still to be determined.

2.7 Conclusion

IP and ATM both have valuable features to suggest they become the technology of choice for telecom operators. However there is a common belief in the telecom community that NGN will be made of all-IP multi-service networks ([Eri01b]).
The migration path is still not clear, and, depending on the existing infrastructure, customers needs, and development strategy, the path to all-IP networks is likely to go through an intermediate phase, where both technologies will be used. As described in [SUOS01], design of multi-protocol routers may be needed to enable the use of hybrids networks.

Chapter 3

Network Simulation

Preliminary work on this project suggested that a network simulator was needed to provide the expected estimates of bandwidth requirements. This chapter provides an overview of network simulation, with descriptions of the main simulators used, and the different types of simulation possible. It also addresses the main concerns rising when building a simulator.

3.1 Simulation in Network Research

As outlined in [BBE+99], network usage is evolving, and researchers need tools to test possible evolution scenarios: the protocols and standards are constantly changing, as well as the usage of the existing network. The reasons for using simulation in general also apply for network research:

- Modelling: when no applicable mathematical model of the system exits, or can be evaluated, the simulation provides a numeric estimate of the characteristics of the system. The goal may be to simulate reality, or to build a model and simulate it.

- Planning: Simulation is used to review several options before deciding which one to use. It's easy to simulate traffic growth to test scalability of a system.

- The use of real testbeds is limited: usually the cost increases with the size, making simulation of wide-area networks difficult. Moreover reconfiguration of the system is difficult and it's impossible to reproduce exactly events showing a random behaviour.

3.2 Network Simulators Overview

3.2.1 Simulation Abstraction Level

Several abstraction levels are used in network simulation. The next sections describes the main ones, while a graphical overview is given on figure 3.1.

Packet based simulation

In packet-based simulation, each packet[1] transmitted on the network generates one event.
This method is the closest to real network behaviour. New protocols can be implemented directly from their specification. However, the number of events generated can be very high and make the problem uncomputable on major systems.
Nevertheless, event-driven packet-based simulation is the most common abstraction used. [KS95] is an example of the use of packet-level simulation to simulate an ATM network.

Fluid simulation

To overcome the computational barrier implicit in the previous methodology, fluid simulation uses a different event model. Instead of generating one event for every packet, the system is considered at an higher level. Data transmissions are considered as flows which are described by their bit rate. The only events being considered are the change of bit-rate for a given flow. A study of fluid simulation of ATM networks

[1]The term *packet* is usually used for IP traffic, while the term *cell* is used in ATM traffic. However *packet* will be used in this documents for both protocols.

is given in [KSCK96].

This method has two main drawbacks. The higher level of abstraction may need some preliminary mathematical study before implementing a traffic type or a network policy. The second drawback is that interactions between flows may increase the amount of events generated drastically. This increase in the number of events caused by interactions is called *ripple effect* in [LFG+01] and [KS95]. Liu, Figueiredo, Gua, Kurose and Towsley showed in [LFG+01] that, for scheduling policies such as single FIFO (First In - First Out) queueing, even on simple and small networks, more events may be generated for fluid simulations than for a packet-based simulations. However, for policies involving less interactions, such as GPS (Generalized Processor Sharing), fluid simulation is still less expensive in computational power. It should also be noted that doubling the bitrate of a source will cause the number of events in packet-based simulation to double as well, while it will not have a significant impact on a fluid simulation.

Hybrid model

Some simulators rely on a time-driven system rather than an event-driven one. Such simulations are discrete: the time is split into small regular intervals. The system state evolves at given time steps with this timescale influencing the precision of the results. This differs significantly from the continuous simulators described above.

Time-Stepped Hybrid Simulation (TSHS) [GGT00] is packet-based: packets emitted during one time-step are grouped and assumed to be evenly spaced during this period of time. This is called *packet smoothing*. [YG99] gives an overview of the performance and precision of a time-driven fluid simulator: flows are fluid, however, the rate remains constant during one time step.

The main advantage of time-driven simulation is the easy use of parallel architectures: all processors can easily agree on checkpoints, namely at the end of every time-step. Parallel algorithms can drastically increase the performance of such simu-

lator.

Mixed mode

Other work ([YMTB01] has been done to extend fluid-modelling analysis to packet-based simulation. The purpose of this method is to simulate some parts of the system with a fluid flow-based analytical mode, and others with a discrete packet-based event simulator, depending on which model is more appropriate to simulate a part of the network. The simulator needs to provide the interface for translating flow model to packet arrival, and the other way. The performance is increased when the flow model is used to analyse sections of the network with heavy traffic, while a finner grained packet-based simulation is used to analyse other parts.

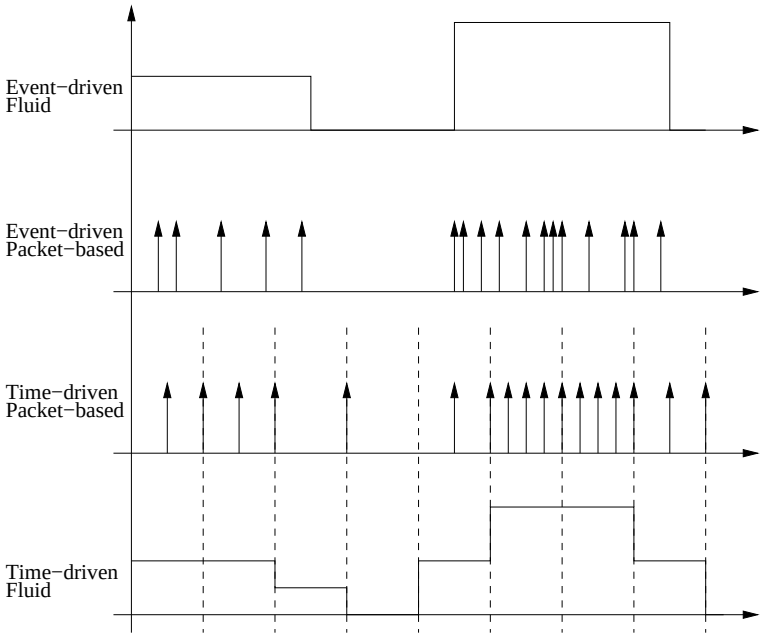


Figure 3.1: Network simulation models

3.2.2 Main Network Simulators

This section is a list of the most important network simulators. This list has been gathered from several reviews, the most important being [Cha99] and [BBE+99]. This list is not exhaustive, as many network simulators have been developed over the past 15 years.

NS2

ns2 is part of the VINT (Virtual Inter-Network Testbed) project. It is a complete network simulator, designed for test and validation of protocols. Written in C++, with an interface in a language derived from Tcl, ns2 is freely available. A graphical interface for analysing results is also available. One important feature of ns2 is the ability to inject live traffic in the simulation. With a large library of protocols, ns2 has been used widely, to investigate TCP behaviour, router queueing policies, multicast transport, multimedia, wireless networking, application-level protocol, etc. A detailed review of ns2 can be found in [BEF+00].

PARSEC (GloMoSim)

PARSEC is developed by the Parallel Computing Laboratory at UCLA, as an extension of the C programming language. The source code can be compiled for different parallel execution protocols, allowing very fast execution time.
Released in 1999, GloMoSim is a Parsec network simulation library designed to simulate wireless and ad hoc networks. It has since been extended by QualSim to handle both wireless and wired networks.

OPNET

Founded in 1986, OPNET is a commercial product targeted in planning and performance optimisation in communication networks. It is a discrete event network simulator written in C. A detailed review of OPNET is given in [Cha99].

MaRS

MaRS (Maryland Routing Simulator), from the University of Maryland, is a discrete-event simulator designed to evaluate and to compare network routing algorithms. The parameters are the actual network topology, traffic sources and routing algorithms. MaRS has been used to evaluate and compare several next-hop routing or link-state algorithms.

CLASS

CLASS stands for ConnectionLess ATM Services Simulator. It's a time-driven, synchronous simulator written in C. [MBD+95] provides a detailed review of the implementation and features of CLASS.

REAL

REAL provides a simulator interface to study the dynamic behaviour of flows and congestion control schemes in packet-switched data networks. The simulation scenario is input using a text file representing the network. Modules are provided to emulate well-known flow control protocols and source types.

INSANE

Written in C++, INSANE (Internet Simulated ATM Networking Environment) is a network simulator designed to test various IP-over-ATM algorithms with realistic traffic loads derived from empirical traffic measurements. Internet protocols are supported, including large subsets of IP, TCP, and UDP.

3.2.3 Alternatives to Network Simulators

Mathematical Modelling

Mathematical software tools, such as Matlab, have been successfully used to model networks elements behaviour. Usually, the overall system is too complex to be anal-

ysed with such tool. However, Matlab may be used to generate code to model a part of a system, which is then integrated into the simulator.

Petri Networks

Petri networks were first developed by Carl Adam Petri in 1962. Petri networks represent a system using a state diagram, along with tokens. As stated in [Mur89] by Murata, Petri networks are a graphical and mathematical tool "*for describing and studying information processing systems that are characterised as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic*".
Some work have been done ([KK01], [MBGT97]) regarding the use of Petri nets for modelling ATM traffic.

3.2.4 Summary

A wide range of tools have been developed to study and analyse networks. Each of these tool has a set of targeted features, there is no general network simulator. Choices and trade-offs have to be made regarding granularity, scalability, and specialisation of the network simulator.

3.3 Technical Aspects

In this section, we will cover the major concerns to be addressed when writing a simulator. This considerations, among many others, may be found in [LK00].

3.3.1 Future Event Set Management

An event-driven simulator has to maintain a list of future events. The management of these events forms a significant part of the processing overhead in such a simulator. [MS81] and [LK00] provide an overview of such algorithms:

- Linked list: this is the simplest management scheme to implement, but the

performance is usually poor, because the insertion of a new event has to be made by linear search. However, retrieving the first element (i.e. the next event at a current time) is fast.

- Indexed list structures: maintaining a set of intermediate pointers can speed up the process of inserting a new event at the right place in the list. The maintenance of these pointers does not usually had a lot of overhead on the computational effort. At first the event to be inserted is compared to the intermediate pointers, then a linear search is started from the appropriate pointer.

- Other data structures: non-linear structures such as binary trees and heaps may be used. These structures may be complex to implement, and their efficiency depends on the distribution of the event set, even if the performance are better than the previous method in general.

These algorithms are not fundamentally different from the well-known sort algorithms. However, the distribution across time of the event set has an impact on the performances, and has to be taken into consideration.

3.3.2 Random Number Generation

[LK00] emphasises the importance of randomness in simulation. The goal is to provide a realistic and accurate numeric estimation of a system, and as most elements involve random behaviour, a random number generator is needed. However, the final simulation results must be obtained by running several simulations using different sequences of random numbers. These sequences must be truly random, and be uncorrelated with each other.
The random variables are generated in two steps:

- a pseudo-random number generator is used to generated one or more uniform variables over $[0, 1]$. The main characteristic of the pseudo-random number generators is that they are deterministic and periodic: the same initial value

(seed) will produce the same results, and after a certain number of generations, the generator will be re-initialised by producing the initial seed. This may seem to be a major drawback for these generators. However, this has been studied in depth, and it has been proved that, provided the period is very long, the sequence is random to all intends and purposes.

- the variable(s) generated are then used to produced a random variate corresponding to the appropriate random distribution. The complexity of this generation and the number of variates from $\mathcal{U}_{[0,1]}$ needed may vary, depending on the complexity of the distribution.

3.4 Conclusion

This chapter provides an overview of the preliminary work carried out prior to the implementation of the network simulator layer of the planning tool. We reviewed the motivation for writing a simulator, along with details of how network simulators are implemented. We also looked at existing network simulators.
The actually software development is described in the next three chapters. Chapter 6 focuses on the network simulator implementation in the planning tool.

Chapter 4

Design & Implementation

This chapter details the work done in the first phase of the project: specification of the software to be implemented, and design of the core parts of the tool.
A detailed view of the re-implementation of [Hov01] and the actual implementation of the network simulator will be studied in the following two chapters.

4.1 Introduction

4.1.1 Specifications

The tool was developed in two phases:

Step 1 Design and implementation of the network model (section 4.3); implementation of algorithms to compute network bandwidth usage.
The algorithms are described in the next section, while a detailed analysis is given in section 7.2.4.

Step 2 Implementation of a network simulator on top of the network structure developed in the first step.
This additional layer was needed to provide additional metrics, which could not be computed using the first design, due to the lack of provision for modelling the dynamic behaviour of the network.

For the simulator (step 2), metrics are provided to measure the following:

- Bandwidth used (bitrate),
- Packet loss ratio,
- Delay,
- Maximum bandwidth used for a link.

Measurements of the continuous evolution of the bitrate through time are available, along with the average value for all measurements of interest, which is produced at the end of the simulation. The tool is able to launch several simulations varying one or several parameters (e.g. buffer size, switch service rate). This allows for easy study of the impact of these parameters on the system behaviour. The random number sequence used will be generated with the same seed across all of these simulations to allow for comparison. As stated in [LK00], one should not rely on one simulation. Several simulations should be used to ensure that the average values are converging.

The C++ programming language was used. As speed was a major drawback of the previous implementation, a special effort was made to optimise the algorithms used in this tool. Speed of the program was considered when making choices in the design phase, and during the implementation. However, some non-optimal solutions have been implemented because of the lack of time. For example, the support for parallel execution has been neglected because of its complexity, in spite of the fact that this technique has a very important effect on the execution time.

4.1.2 Common Aspects

In the next sections, we will describe the aspects which are common to the main parts of the tool.

- *Routing*: The routing is static and it is computed before the computation of

the bandwidth or the simulation. This is done using a simple shortest-path algorithm.

- *Network model*: The network infrastructure is described in same way for both stages of the implementation. The information which is provided to the simulator is more complex, as both source model and switch technology impact on the results.

- *Scripting language*: The interface instantiates objects representing the network elements and agents. An extended version has been developed to cover the subclasses needed for the second version of the tool.

- *Environment*: All the objects are stored in a common environment. This allows for ease of instantiation, as well as common object reference, in the input interface.

4.2 Routing

The routing is done using a shortest-path algorithm. The links can be weighted either by their respective delays or equally.
This algorithm is complex ($O(S^3)$, where S is the number of nodes). However, it computes all the paths, which are needed for the bandwidth requirement computation. The main drawback of this method is that it rules out the possibility of enabling dynamic routing.

4.2.1 Notation

The following notation is used in the next section:

- S is the number of nodes, and M the number of links in the environment,

- $N = (n_1, n_2, \cdots, n_S)$ is the list of nodes,

- $L = (l_1, l_2, \cdots, l_M)$ is the list of links,

- for $n_i, n_j \in N$, $P(n_i, n_j)$ is the path from node n_i to node n_j, and $d(n_i, n_j)$ is the weighted length of this path,
- for a link $l \in L$, $n_a(l)$ and $n_b(l)$ are the two nodes at each end of the link (we assume that $n_a(l) \neq n_b(l)$.

4.2.2 Algorithm

The routing algorithm involves three steps. The first two initialise the algorithm. Step one initialises all distances to infinity, except for one node to itself where the distance is 0. There are no paths specified. The second step initialises the distance and the path for each direct connections.
Finally, the third step iterates through all the paths to check if the use of an intermediate node can shorten the path. If it does, the path is modified accordingly.

1. `for all` $n_i \in N$
  `for all` $n_j \in N$
    `if` $i = j$, $\begin{cases} P(n_i, n_j) = [\emptyset], P(n_j, n_i) = [\emptyset] \\ d(n_i, n_j) = 0, d(n_j, n_i) = 0 \end{cases}$
    `else` $d(n_i, n_j) = \infty$, $d(n_j, n_i) = \infty$

2. `for all` $l \in L$
  $d(n_a(l), n_b(l)) = W(l)$,
  $d(n_b(l), n_a(l)) = W(l)$,
  $P(n_a(l), n_b(l)) = [l]$,
  $P(n_b(l), n_a(l)) = [l]$.

3. `for all` $n_k \in N$
  `for all` $n_i \in N$
    `for all` $n_j \in N$
      `if` $d(n_i, n_j) > d(n_i, n_k) + d(n_k, n_j)$ `then`
        $P(n_i, n_j) = P(n_i, n_k) + P(n_k, n_j)$

4.3 Network Model

4.3.1 Model

The network is represented as a set of nodes bounded by links. The agents are the sources, switches (or routers), and sinks on the network. At least one agent has to be mapped to every node. Only one switch can be mapped to a given node. Subclasses of `Source` represent different traffic models, while subclasses of `Switch` represent different scheduling policies. Note that the subclasses of `Source` and `Switch` are not relevant for the bandwidth calculation algorithm. Details about the properties of these classes are given in the following section.

4.3.2 Class Hierarchy

Element

- Agent
  - Sink
  - Source
    - SourceCBR
    - SourceMMP
    - SourceOnOff
  - Switch
    - PriorityScheduler
    - QueueMultiplexer
    - SingleFifo
- Link
- Network
- Node

- Value
  - Const
  - Var

4.3.3 Class Diagram

The class diagram is given on figure 4.1. This diagram means that a network is composed of nodes and links. A link is associated with two different nodes. Traffic agents (sources, sinks and switches) have to be attached to each nodes.

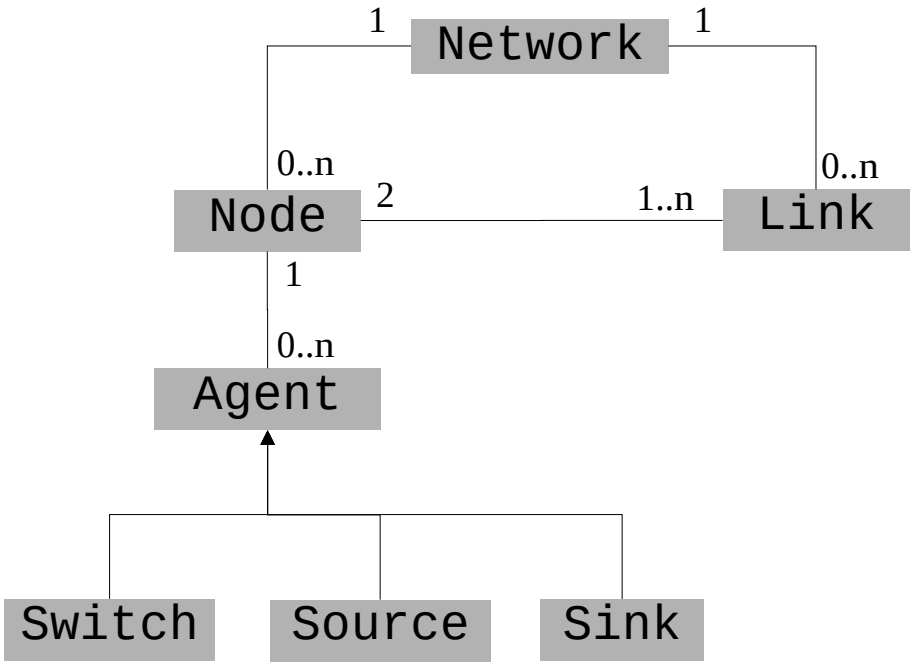


Figure 4.1: Class Diagram

4.3.4 Example

A sample network is shown on figure 4.2. It consists of four nodes, $N = (n_1, n_2, n_3, n_4)$, and 3 links, $L = (l_a, l_b, l_c)$. Two sources, `source1` and `source2` are mapped to nodes n_1 and n_2 respectively. The traffic from these sources is going through the switch `switchX` mapped to n_3, to the sink `sink_all` mapped to n_4.

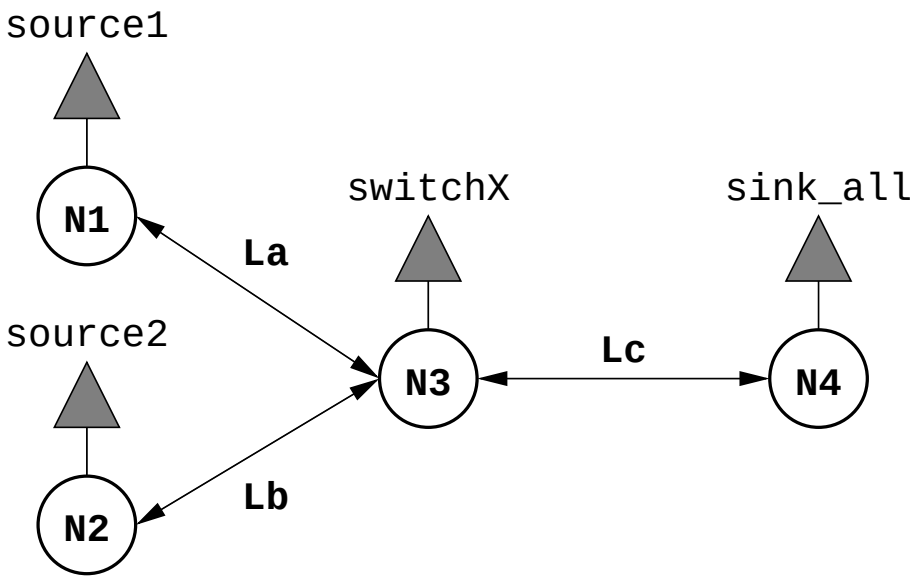


Figure 4.2: Example Network

4.4 Interface

The interface is text-based. A graphical interface had been planned in the initial proposal, but efforts have concentrated on improving the tool's features. However, the system is designed to allow an easy implementation of a GUI on top of the current script based interface.

4.4.1 Input Scripting language

Syntax

Scripts are used to provide input to the simulator. The core set of command is:

- `new <class> <object_name> [<param1> ...]`: create a new instance of `class` named `object_name`. The parameters are optional or mandatory, depending on the class instantiated.
- `set <object_name> <property> <value>`: set `property` of the object named `object_name` to `value`.
- `include <file_name>`: process the contents of `file_name` as a script, and then

process the rest of the calling script file. This can be done recursively. This feature allows easier configuration of very similar network scenarios.

- `compute_bw`: compute the bandwidth required on every link on the given network.
- `run_sim`: start the simulation.
- `print_env`: print the environment (debug only).

Lines beginning with `#` are considered as comments. If the tool is called without any parameter file, a shell will be open and the user will be prompted for commands line by line.
Complete syntax and examples are given in sections 5.3 and 6.2.

Example

This is the script used to instantiate the objects representing the network used as an example in section 4.3.4 (figure 4.2).

```
# network infrastructure
   new node N1
   new node N2
   new node N3
   new node N3
   new link La N1 N3
   new link Lb N2 N3
   new link Lc N3 N4

# traffic agents
   new sink sink_all N4
```

```
    new switch switchX N3

    new const bitrate 100.0

    new source source1 N1
    set source1 rate bitrate
    set source1 dest sink_all

    new source source2 N2
    set source2 rate bitrate
    set source2 dest sink_all

# bandwidth
    compute_bw
```

4.4.2 Output

Different types of output files are generated, depending on the functionality used:

- Bandwidth usage: a file containing a tab-separated array is produced. For every link, the following data is given: link name, average bitrate, bandwidth required on PSTN networks, bandwidth required on ATM networks, difference in percentage. An example is given in section 7.2, table 7.2.

- Simulation: The evolution of some network element values is stored in a file in Gnuplot format. Only the values specified in the input script will be plotted. The main drawback of this strategy is that the simulation needs to be re-run if another value needs to be studied. On the other hand, writing this files adds a lot of overhead on computation, and plotting all the values is certainly not a solution.

4.5 Environment

4.5.1 Hash Table Structure

Every object instantiated by a script has an unique name. For ease of retrieval, the environment is represented using an hash table, with the object name used as a key. The hash value is computed using this function, to ensure maximum dispersion of the values across the table:

```
int hash_value (string name) {
     int ret = 0;
     for (unsigned int i=0; i<name.length(); i++) {
          ret += name[i] * name[i] * (i+1);
     }
     return ret % ENV_DEFAULT_SIZE;
}
```

4.5.2 Variables

For research purposes, the ability to run several simulations changing a few parameters, and keeping the same set of random seeds for the random-driven source is vital.
To allow for study of the impact of one or several parameters, the following strategy is used. All the numeric object properties, which on the lowest level as considered as C++ `double`, are instantiations of one of the subclasses of the class `Value`. There is two subclasses of `Value`: `Const` and `Var`.

Subclass `Const`

This is used to represent a value which will not change across the simulations. However, it provides an easy way to set multiple parameters to the same value, and

to update this script (e.g. the delays associated with all the links). When a value is expected by the script interpreter, a literal number can be inputed. In this case, a `Const` object is instantiated. Its name is the string representation of the number.

Subclass `Var`

This subclass is used to run several simulations with the object of this class varying its value. The syntax to instantiate such objects in the script is: `new var <name> <init> <end> <increment>`. The value is started at `<init>`, incremented by `<increment>`, while the value is equal to or lower than `<end>`.
If multiple `Var` objects are instantiated, all the possible combinations will be run.

4.6 Chapter Summary

In this section we have reviewed all the abstraction levels and interfaces used in the current implementation. The common features are: routing, network infrastructure model, script interpretation, and variables environment. All these features have been developed for the bandwidth computation, and have been extended if needed for the network simulator.

Chapter 5

Bandwidth Computation

5.1 Description

In this chapter, we will review the features of the re-implementation of Hovland's work. The results are an estimation of the gain in bandwidth requirements obtained with the migration from a telephony PSTN-based network to an ATM-based network. The next sections describe the source model and the algorithms used for bandwidth gain computation. A detailed analysis is given in section 7.2.4.

5.2 Traffic Source Model

We are considering three different types of traffic: High Quality voice, Low Quality voice, and Video. The traffic is described using the ATM specification. This kind of traffic is bursty: during some time, the bitrate is Peak Cell Rate (PCR), and then is reduced to Sustainable Cell Rate (SCR). Such traffic is described on figure 5.1, and the data used, provided by Eircom, is given in table 5.1.

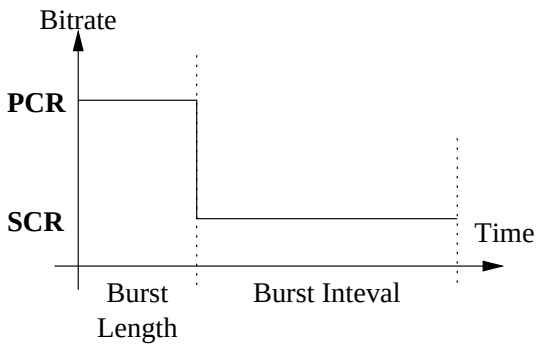


Figure 5.1: Traffic types

Traffic Type	PCR kb/s	SCR kb/s	Burst Length	Burst Interval	AAL
HQ Voice	64	64	*	*	AAL1
LQ Voice	64	16	30	300	AAL2
Video	256	32	30	300	AAL2

Table 5.1: Traffic Types

5.3 Input Script

5.3.1 Syntax

To set up the configuration, the network topology needs to be defined, using `Node` and `Link` objects. The syntax to define those is:

- `new node <name>`,
- `new link <name> <node1> <node2>`.

Then agents are mapped to the nodes. For bandwidth computation, we do not consider traffic policies, so all the nodes are implicitly used as a switch, there is no need to instantiate `Switch` objects. `Source` and `Sink` objects are defined by:

- `new source <name> <node>`,
  `set <name> rate <value>`,
  `set <name> dest <sink>`,

- `new sink <name> <node>`.

Several sources can have the same sink as a destination.

Some choices have been made regarding the parameters which should be included in the instantiation of an object (command `new`) or should be set after (command `set`). In an early version of the scripting language, all the parameters had to be set. However, this solution tends to make the scripts long. On the other hand, the numbers of parameters on the `new` command line has to stay small, as too many parameters tends to be confusing. Some parameters are mandatory on the `new` command line, while others are to be `set`; the general criteria is to include in the command line parameters which are used in both versions of the tool.

5.3.2 Example

The example given in section 4.4.1 is actually a valid script for bandwidth computation:

```
# network infrastructure
   new node N1
   new node N2
   new node N3
   new node N3
   new link La N1 N3
   new link Lb N2 N3
   new link Lc N3 N4


# traffic agents
   new sink sink_all N4


   new switch switchX N3
```

```
    new const bitrate 100.0

    new source source1 N1
    set source1 rate bitrate
    set source1 dest sink_all

    new source source2 N2
    set source2 rate bitrate
    set source2 dest sink_all

# bandwidth
    compute_bw
```

5.4 Algorithm

The first step computes the exact traffic mix on all links, while the second step computes the bandwidth needed for PSTN and ATM-based networks.

Step 1 for all sources
  for all links in path (source, sink)
    add (traffic type, rate)

Step 2 for all links
  compute bandwidth (PSTN)
  compute bandwidth (ATM)

Chapter 6

Network Simulator

6.1 Implementation

This chapter describes the implementation of a simple network simulator. This functions on top of the previously described tool.
A fluid, event-driven simulator has been used. This type of simulator has been chosen because it provides a good estimation of the metrics needed, it is usually faster than packet-based simulation, and because we only need estimates rather than exact values. This chapter gives an overview of extensions to the existing program, and of the key algorithms used at simulation run-time.

6.2 Scripting Language Extension

The scripting language has been extended to integrate the new features.

Network Agents

The `new` commands for `source` and `switch` have been updated. As the `sink` class is passive and used only to collect data, there was no need to alter this.
`new <source_type> <name> [<parameters>...] <rate>`. There is three different source types, `source_cbr`, `source_mmp` and `source_onoff`. These types correspond

to the three traffic source models described in section 6.4. The command lines are:

- `new source_cbr <name> <bitrate>`,
- `new source_mmp <name> <lambda> <mu> <off_rate> <on_rate>`,
- `new source_type <name> <lambda> <mu> <N> <beta>`

For all the sources, two additional parameters have to be set: `start` and `stop`, with the starting and stopping times for the source. When a priority scheduler is involved, the parameter `priority` has to be set.
For the subclasses of `Switch` described in section 6.5, the syntax is:

- `new <switch_type> <name> <service_rate> <buffer_size>`,

where `<switch_type>` must be `fifo`, `gps`, or `prio`.

Output Control

The output is simple text files which can be used as input for the Gnuplot program. As writing to these files take a significant time, the user needs to specify precisely which graph should be plotted.
There is two different types of graphs generated by the tool: evolution of bitrate of an agent through time, and loss and delay for a source, as a function of a system parameter.
The first option allows multiple agent bitrates to be plotted on the same graph. This is done using the class `monitor` and the command `plot` as follow:

```
new monitor <name>
plot <agent> <monitor>
```

The rate value plotted depends on the agent type:

- `source`: output bitrate,
- `switch`: output bitrate (sum of all flows),
- `sink`: received bitrate.

The metrics for the flows (identified by a source) are produced using:
`graph <source_name> <x_var> [<y_var>]`,
where `x_var` and `y_var` are the name of two `Var` objects. `y_var` is optional. If it is used, a 3D graph will be plotted.

6.3 Events Scheduling

This section outlines the events scheduling algorithm, a key part of the network simulator.

6.3.1 `Event` Class

Class Hierarchy

Event

- BufferEvent
- DepartureEvent
- FlowEvent
- SourceEvent

Description

Four different types of event are implemented:

- Source state: This event represents a change in a source bitrate. Only one event of this type is scheduled for each source, to keep the event list small.
- Flow bitrate: represents a change in a flow arrival bitrate at one node.
- Departure bitrate: represents a change in a flow departure bitrate at one node.

- Buffer state: as the filling status of the buffer influences the behaviour of the system, the simulator needs to schedule the changes on the buffer status.

The event propagation mechanism is demonstrated in next section.

6.3.2 Example

The network described in figure 6.1 is used in this example.

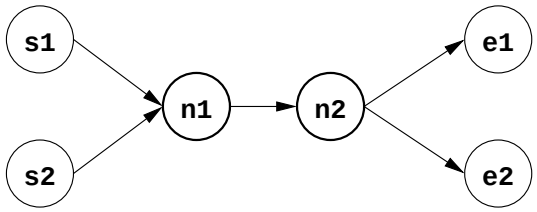


Figure 6.1: Example Network

This network consists in two sources, `s1` and `s2`, sending data respectively to the sinks `e1` and `e2`. All the traffic goes through the switches `n1` and `n2`. All the links have the same bandwidth b and propagation time θ. `s1` sends cells between t_0 and t_2, and `s1` between t_1 and t_3. We have $t_0 < t_1 < t_2 < t_3$ and $t_i >> \theta$, with $i \in \{0, 1, 2, 3\}$. The second assumption is not needed by the simulator, but it makes this example clearer. During their transmission time, the two sources are sending data at the same bit rate $0.6b$. This value is used to outline the buffer state management: when `s1` and `s2` are sending data together, between t_1 and t_2, the dimension of the link `n1-n2` is too small to handle a bitrate of $1.2b$.

Figure 6.2 represents the relationships between events. Arrows are representing events, a dashed line between events A and B, with $t_A < t_B$, means B is spawned by A; `2` at the base of an arrow means two simultaneous events. Uncommented events represent a change in bit rate in the relevant flow.

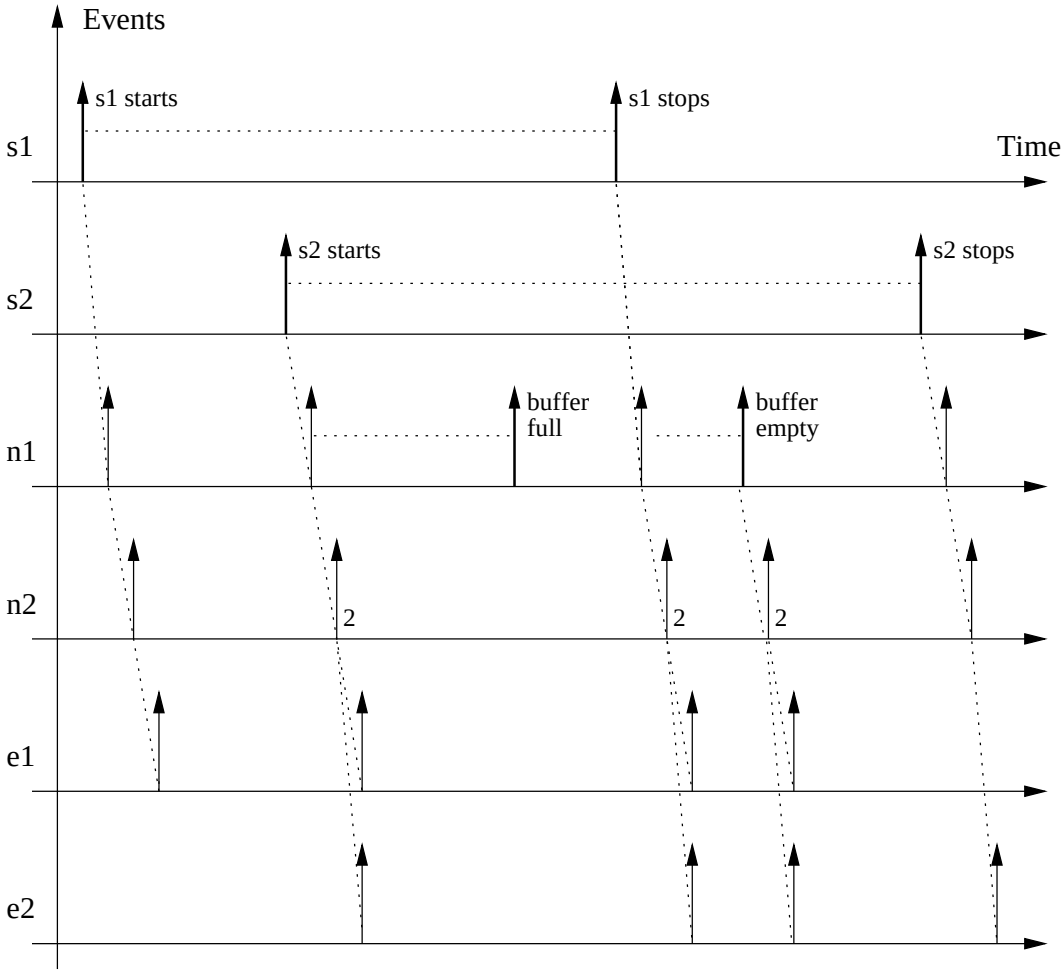


Figure 6.2: Events Relationships

6.3.3 Future Events List Management

The strategy described in the previous section allows for the event list to stay relatively small. At any given time, there is usually one event per source for the next state transition, and at most one more event for the propagation of the last change through the path. At node level, the next change in the status of the buffer (full or empty) may be scheduled if necessary.

Maintaining a small and relatively constant event list over time is good for performance and memory usage. As most of events only spawn one other event of the same type, memory used for this event can be recycled, instead of being freed and reallocated. The event list needs to be, and remain sorted. Minimising its size is quite important, even if there are algorithms which are better than linear search for this task.

In this implementation, three pointers to relevant elements of the list are kept up to date: the first element, the last element and the last inserted element. When an element has to be inserted or updated, its timestamp is compared to these three values. It may be inserted before the first element, after the last one, or in between using a linear search starting from the first element or from the last inserted element.

6.4 Traffic Models

Every source has a traffic model attached. All the models used in this simulator should be described as a series of constant bitrate traffic flows. While the source is evolving through states, the bitrate during an individual state should stay constant. However, the length of the state and the bitrate associated may be determined by random values. Figure 6.3 represents a source going through five different states during the simulation time.

This section contains a detailed list of the available traffic models. The following graphs are used to show the abilities of the tool to produce graphs of bitrate as a

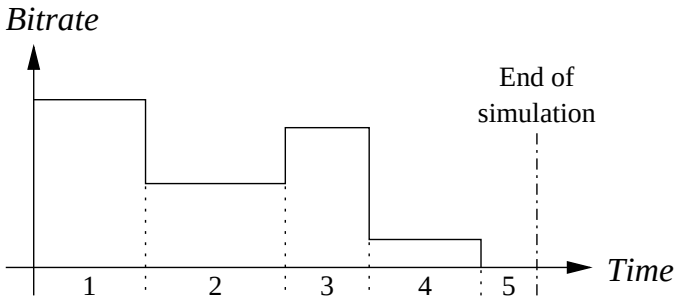


Figure 6.3: Example Source Bit Rate

function of time. This feature has been very useful for debugging purposes, but the significant results are usually the average values over one simulation.
The different traffic models have been selected from several surveys ([JMW96], [MM99], [RK96]). The traffic models used were chosen because they are easy to implement, and are sufficient to simulate various traffic types.

6.4.1 Constant Bit Rate

This model is used to simulate constant traffic, such as that described by the CBR traffic class of ATM. A sample is given in figure 6.4. This traffic was not used much during the detailed analysis phase of the project (chapter 7), but it was used early in the implementation process for debugging purposes, as it represents the simplest source, with only one state. This source model is only defined by the bitrate β.

6.4.2 Markov-Modulated Rate Process

For the sources of this type, the underlying mechanism is a Markov chain with N states. The bitrate at any given time is the state index multiplied by the bitrate per state value, β: at state n, the bitrate of the source is $n \times \beta$.
λ and μ represent the parameters of the exponential probability law, which define respectively the probability of moving from state n to $n+1$, and from state n to $n-1$. The Markov-Modulated Rate Process is described on figure 6.5, while figure

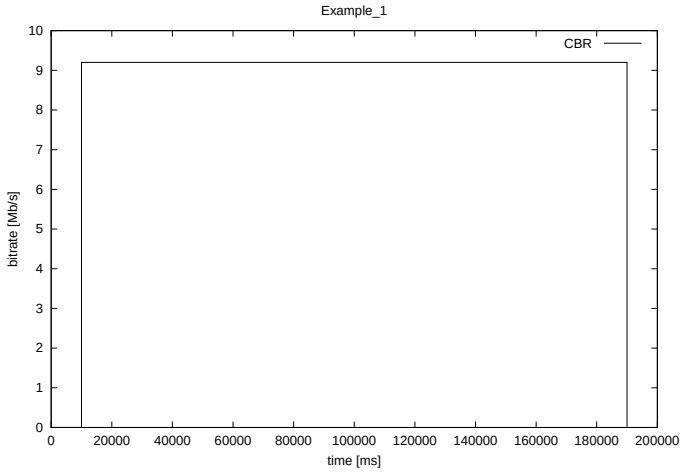


Figure 6.4: Example: Constant Bit Rate

6.6 gives a sample bitrate output of such source.

So, as described in figure 6.5, this model is defined by (N, β, λ, μ).

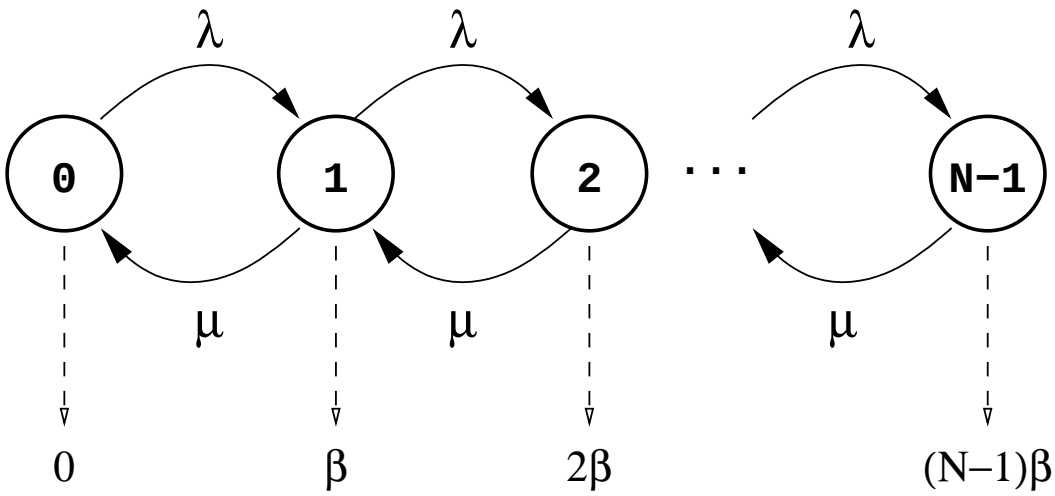


Figure 6.5: Markov-Modulated Rate Process

This model is used to represent common over-IP protocols (TCP, FTP), and groups of telephone calls. They are appropriate to model IP Telephony, as described in [And00] and [AAHM01]. Kesidis and Walrand, in [KW93] and [KWC93], have studied the properties of Markov multi-class fluid sources. A sample output is given on figure 6.6.

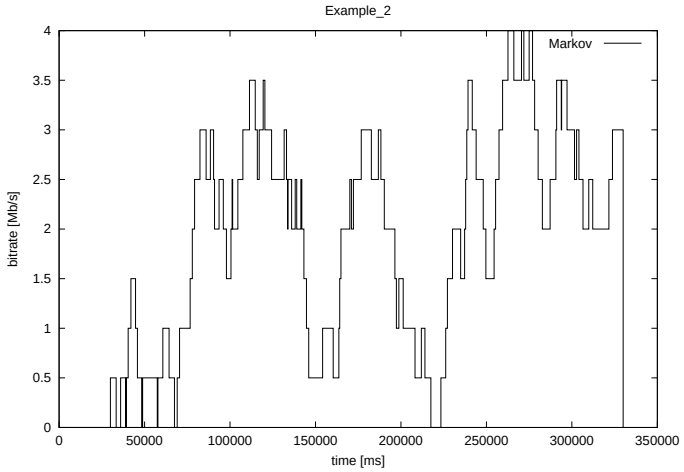


Figure 6.6: Example: Markov-Modulated Process

6.4.3 On-Off Sources

This traffic type is actually a Markov-modulated On-Off source. It is similar to the previous one, except that there are only 2 states, and the rate is not proportional to the state index. In this case, state `OFF` is associated with bitrate β_0, and state `ON` with β_1.

This model is defined by $(\beta_0, \beta_1, \lambda, \mu)$, as described on figure 6.7.

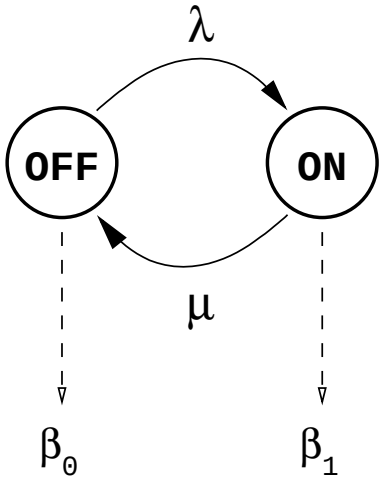


Figure 6.7: Markov-Modulated On-Off

According to [HKS99], it can be used to represent Web traffic, or VBR class of ATM. A sample output is given on figure 6.8.

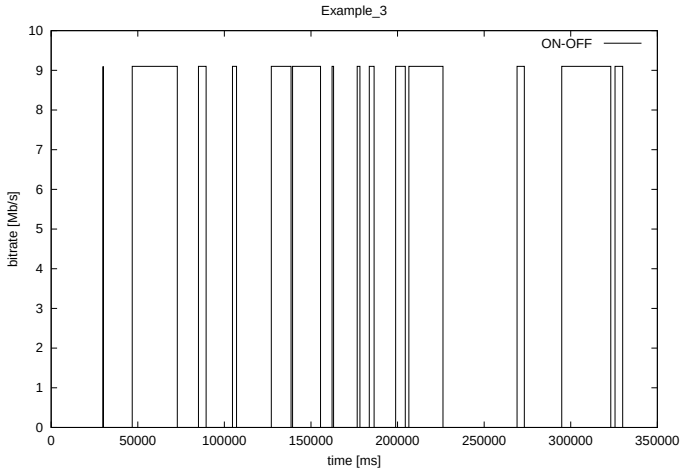


Figure 6.8: Example: On-Off Sources

6.5 Scheduling Policy

On the router/switch side, three different traffic management policies have been implemented. According to [MKD01] and [PD00], these policies represent the most commonly used. These are: FIFO Queue Scheduler, Generalized Processor Sharing (GPS) and Priority Scheduler.
For implementation of the simulator, two aspects have to be considered:

- On the queue level, we need to know how a change in the arrival process affects the departure process. This is described with the FIFO queue scheduler.
- If the service of a FIFO queue can be modified, which is the case for GPS and priority scheduler, we need to know how a change in the departure or arrival rates in a queue affects the distribution of service rates.

6.5.1 FIFO Queue Scheduler

This scheduler is the most common one, and the next two scheduler types are basically using several FIFO queues, with some policies regarding the share of service rate they may receive.

Notations

We consider a FIFO queue, with a service rate c, considered as constant and non zero, shared by N flows. The maximum size of the buffer is B bytes. At any time $t > 0$, for $n \in [1, N]$, we have:

- $a_n(t)$ and $d_n(t)$ are the arrival and departure rates for flow n.
  $a(t) = \sum_{i=1}^{N} a_i(t)$ is the total arrival rate at the queue.
- $X(t)$ is the aggregate buffer occupancy.
- $v(t) = t + \frac{X(t)}{c}$, described as the virtual delay process.
- $l_n(t)$ is the loss rate of flow n.

Departure Process

The departure process of flow n at time t is:

$$d_n(v(t)) = \begin{cases} \frac{a_n(t)}{a(t)} c & \text{if } X(t) > 0, a(t) > 0 \\ \min(a_n(t), c) & \text{else} \end{cases}$$

The loss rate is given by:

$$l_n(t) = 1_{\{X=B\}}(a_n(t) - d_n(t))$$

So, when an arrival event is triggered, a departure event is scheduled some time in the future. The buffer occupancy X is updated to compute the loss and departure rates.

Service Rate Update

To obtain these results, we assumed that the service rate associated with a queue is constant over time. However, as we will see in the next two sections, this service rate may change due to the interaction with other queues at one switch.

For every flow, a list of departure events which have been scheduled but not triggered is maintained. If the service rate is updated, the new bitrates of these events are updated accordingly and the events are rescheduled at another time.
However, when the service rate is updated to 0 Mbit/s, we need to remove all the events linked with this flow from the global event list, and store the current time and current bitrate. When the service rate is put back to a non null value, the appropriate time and bitrate for these events are computed, and the events are inserted back to the event list.
This strategy enables an implicit declaration of the buffer occupancy, as the only explicit value is the size of the aggregate buffer.

6.5.2 Generalized Processor Sharing

For a GPS node, the bandwidth available is divided between a number of FIFO queues, say N. Each queue is characterised by its parameter. The partitioning parameters $(\phi_1, \phi_2, \cdots, \phi_N)$ are defined such as $\forall i \in \{1, 2, \cdots, N\}, \phi_i > 0$ and $\sum_{i=1}^{N} \phi_i \leq 1$. We define A_i such as $A_i = 1$ if the queue i is active, and $A_i = 0$ else. Note that $A_i = 1_{\{X_i>0\}} \times 1_{\{a_i>0\}}$. We also define $\bar{A}_i = A_i - 1$. We assume in the rest of this section that there is, at any given time, at least one queue i such as $A_i = 1$. This is false only when the entire queue scheduler is idle, a situation that is of no interest for our research.
The normal service rate of a queue is given by:

$$c_n = \frac{\phi_n}{\sum_{i=1}^{N} \phi_i} A_n c$$

However, when one of the queues is idle, its share of the total service rate is divided among the other queues. The strategy to distribute unused bandwidth can vary from one queue multiplexer to another. The general equation for a non-idling queue multiplexer is:

$$c_n = \left(\frac{\phi_n}{\sum_{i=1}^{N} \phi_i} + \psi_n \right) A_n c$$

where ψ_n represents the share of the idle bandwidth the active queue n will get. $\psi = 0$ represents an idling scheduler.

For GPS, the remaining service rate is distributed according to the weight of each queue:

$$\psi_n = \frac{\phi_n}{\sum_{i=1}^{N} \phi_i A_i} \times \frac{\sum_{i=1}^{N} \phi_i \bar{A}_i}{\sum_{i=1}^{N} \phi_i}$$

Because $\sum_{i=1}^{N} \phi_i \bar{A}_i + \sum_{i=1}^{N} \phi_i A_i = \sum_{i=1}^{N} \phi_i$, we simply have for GPS:

$$c_n = \frac{\phi_n}{\sum_{i=1}^{N} \phi_i A_i} A_n c$$

Other definitions of ψ may be used, and other parameters such as buffer occupancy or input rate can be used to compute how the non-used bandwidth shall be distributed. Only GPS has been implemented, but the implementation of other strategies can be done easily.

Another choice is the way of distributing flows among the queues. This can be done in function of source, destination, next hop, etc.

6.5.3 Priority Scheduler

We consider here the case where all flows are given a priority. All the flows with the same priority class are sharing the same queue. The queues with the highest priorities are served first, and the remaining service rate is given recursively to the queues of lesser priority.

If we have N queues $\{Q_1, Q_2, \cdots, Q_N\}$ such as, for $i \in [1, N-1]$, Q_i has a greater priority than Q_{i+1}, then the service rate is defined recursively by:

$$\begin{cases} c_1 &= 1_{\{X_1>0\}} c + 1_{\{X_1=0\}} \min(a_n, c) \\ \forall n \in [2, N], c_n &= 1_{\{X_n>0\}} \left(c - \sum_{i=1}^{n-1} c_i\right) + 1_{\{X_n=0\}} \min\left(a_n, c - \sum_{i=1}^{n-1} c_i\right) \end{cases}$$

6.6 Delay and Loss Computation

6.6.1 Preliminary Statements

For all flows (represented by the sources), the tool keeps up-to-date values of the date sent, loss, and received rates. Let $s(t)$, $l(t)$ and $r(t)$ represent these values respectively at time $t > 0$. These are computed using the information sent back by both the switches on the data path and the sink associated with the source.

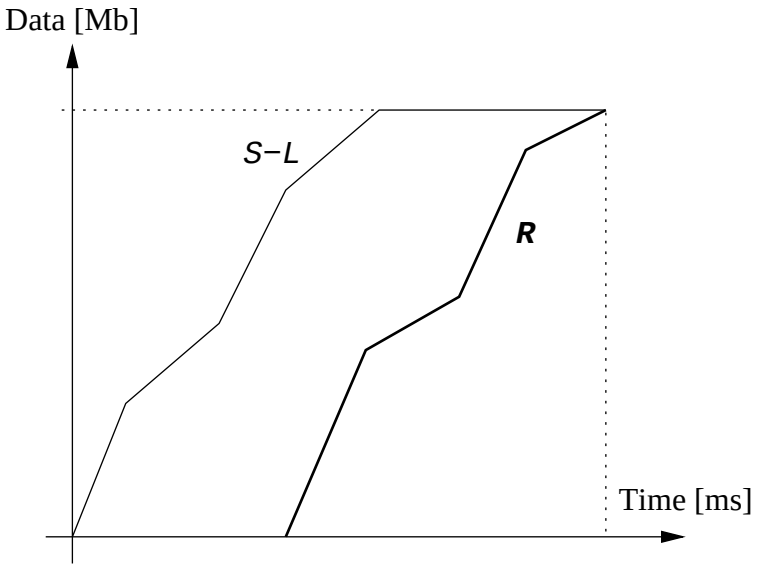


Figure 6.9: Sample flow evolution

The total of data sent, lost and received is also computed, represented by $S(t)$, $L(t)$ and $R(t)$ respectively. Define $\hat{S} = S - L$. See figure 6.9 for example. S, T and R are computed using the following equations (with the example of the data received, R):

- The theoretical value of R is: $R(t) = \int_0^t r(u)\ du$.
- However, because of the fluid simulation model, we know that r is constant over small periods of time. So, there is $N > 0$, $(r_n)_{n\in[0,N]}$, $(t_n)_{n\in[0,N+1]}$ with

$\forall n \in [0, N], r_n > 0, t_n < t_{n+1}$ and $t_0 = 0$, such as:

$$r(t) = \sum_{n=0}^{N} r_n \ 1_{t_n < t < t_{n+1}}$$

- So, R can be computed recursively at every update:

$$\forall i \in [0, N], R(t_{i+1}) = R(t_i) + r_i(t_{i+1} - t_i)$$

- We also need the value $\Omega(t) = \int_0^t R(u)du$. By integrating, we get:

$$\forall i \in [0, N], \Omega(t_{i+1}) = \Omega(t_i) + R(t_i)(t_{i+1} - t_i) + r_i \frac{(t_{i+1} - t_i)^2}{2} \tag{6.1}$$

6.6.2 Computation

Loss

The loss rate is given by:

$$\forall t > 0, \rho(t) = \frac{L(t)}{S(t)}. \tag{6.2}$$

Delay

According to [MKD01], the average end-to-end delay of a connection over $[0, t]$ is:

$$\Delta(t) = \frac{1}{R(t)} \int_0^{R(t)} \left(R^{-1}(u) - \hat{S}^{-1}(u) \right) \ du.$$

However, if at any time $t > 0$, $r(t) = 0$ or $l(t) \geq s(t)$, then respectively R or $\hat{S}$ are not invertible. The problem is that these cases happen:

- In the beginning of the simulation, before the first packet arrives, $r = 0$,
- During the simulation, it is likely to have $l \geq s$, especially when the source does not send anything ($s = 0$), and the data on line is experiencing some loss

$(l > 0)$.

As we are mainly interested in the average delay over the simulation time, we rewrite this equation. Using τ be the total simulation time:

$$\tilde{\Delta}(\tau) = \frac{1}{R(\tau)} \int_0^\tau \left(\hat{S}(t) - R(t) \right) \, dt \tag{6.3}$$

$$= \frac{1}{R(\tau)} \left(\int_0^\tau S(t) \, dt \; - \int_0^\tau L(t) \, dt \; - \int_0^\tau R(t) \, dt \right). \tag{6.4}$$

We have $\Delta(\tau) = \tilde{\Delta}(\tau)$ if $\Delta(\tau)$ is defined (these values are then equal to the area between the two functions on figure 6.9 divided by $R(\tau)$). However, $\tilde{\Delta}$ is always defined, and provides a good approximation of the real value of the delay.
Using the fluid model, we do not have a precise information on the time of transmission of the data as opposed to the packet-based simulation where each packet/event can store a timestamp.
Finally, the value of $\tilde{\Delta}$ is easily computable using the equation 6.1.

6.7 Chapter Summary

In this chapter, we reviewed the integration of a network simulator in the existing program structure, and the main difficulties overcame during the implementation of this simulator: global event list management, traffic source models, scheduling policy models, and finally loss and delay computation.

Chapter 7

Tests & Results

7.1 Introduction

This chapter describes how the features of the planning tool may be used to simulate and study various network topologies. Section 7.2 contains a description of the bandwidth estimation features while section 7.3 focuses on the network simulator.

7.2 Bandwidth Allocation

7.2.1 Metrics Provided

In this section, we study results obtained from the re-implementation of [Hov01]. In particular, we look at the bandwidth savings due to the migration from a PSTN to ATM-based network.

7.2.2 Input Data

Network Infrastructure

The network topology used for the following work was provided by Eircom. It is represented in figure 7.1. The network is centred on five exchange switches, four of these are interconnected through an existing ATM cloud.

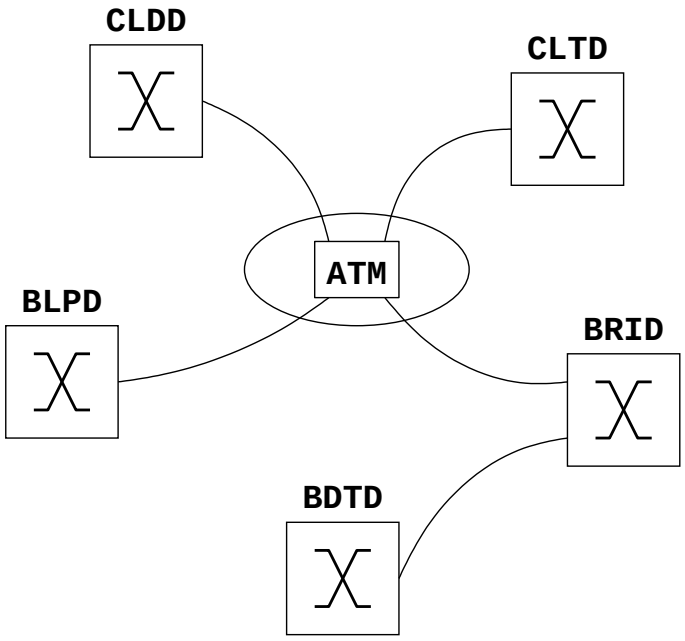


Figure 7.1: Eircom Network

Bandwidth Usage

The data in table 7.1 comes from measurements taken by Eircom on a part of their network. This data was obtained during one busy hour of the day. The units used for the data are Mb/hour.
The amount of data internal to one node (e.g. data from BDTD to BDTD) is not required for the computation of the results. This is because the problem we are considering is the estimation of bandwidth requirements for link dimensioning; the internal communication going through one exchange switch at a lower level is not considered.

Traffic Mix

For this test configuration, we consider that the three traffic types share the bandwidth equally (one third of the total amount of data is generated by sources conforming to each model).

From / To	BDTD	BLPD	BRID	CLDD	CLTD
BDTD	6636.9	1226.552	580.6396	870.1775	493.3056
BLPD	117.596	5756.927	197.0452	467.5476	1381.666
BRID	443.2066	192.7519	4772.021	334.033	132.6762
CLDD	793.9983	520.8473	293.5173	4139.393	212.2588
CLTD	591.2209	1263.141	119.1878	221.7466	1950.31

Table 7.1: Traffic matrix

7.2.3 Results

The results for this configuration are given in table 7.2. The traffic mix is made by an equal amount of data for each traffic type as described above.

Link Name	Average MB/s	PSTN bw MB/s	ATM bw MB/s	Difference %
CLTD to ATM	9.79	28.1	11.62	-58.66
ATM to BRID	11.89	34.3	14.11	-58.86
CLDD to ATM	8.25	23.68	9.79	-58.64
BRID to BDTD	11.35	32.77	13.47	- 58.9
ATM to BLPD	11.92	34.3	14.15	-58.74

Table 7.2: Bandwidth Usage

7.2.4 Analysis

Comparison with the previous implementation

The new implementation of the planning tool gives identical results to that of the previous version. However, the change in the interface, the underlying technology, and the improved implementation of some of the algorithms, contribute to a reduction of several orders of magnitude in the computing time (from about 20s to less than 10ms).

Analysis

For a given traffic type on a given link, let $b(t)$ be the bitrate at time t. Let $b_M = \max_{t>0}(b(t))$ be the maximum of this bitrate, and $\overline{b} = \frac{1}{t}\int_0^t b(t)dt$ the average of

this bitrate during $[0, t]$.

Details of the algorithms used can be found in [Hov01] but, for completeness, we include here the main assumptions for bandwidth requirements for a source of bitrate $b(t)$:

- The bandwidth needed over a PSTN network is b_M, rounded up to the next 64 kbit/s value. Given that the bitrates involved are far greater than this, we can assume that the bandwidth needed is b_M;
- For ATM-based networks, we assume that statistical multiplexing is working at its best, which means that the overall bitrate is distributed such that the bandwidth used is $\overline{b}$. However, ATM cells also carry a significant overhead. This overhead is proportional to the amount of data sent, and represents about 10% (depending on the AAL used, i.e. the type of the traffic) of the data. Let α represent this overhead.

Finally, the proportion of gain in bandwidth is given by :

$$\frac{\alpha\overline{b}}{b_M} - 1$$

For a given link, a weighted average of all the traffic types going through this link gives the figure found in the last column of table 7.2.

This is a rough estimate, based on the assumption that the traffic is distributed so that statistical multiplexing averages it perfectly. This is certainly not the case in real systems. However, the purpose of this tool is to help with link dimensioning, and it does not need to be highly accurate. This method has been approved by people working for Eircom who collaborated with both [Hov01] and this work. And in [CFW94], Courcoubetis, Fouskas and Weber show that, for many stationary sources, the effective bandwidth in ATM networks can be computed using only the source mean rate, its index of dispersion, and the buffer size.

7.3 Network Simulation

In this section we will explore the capabilities of the network simulator. We analyse the results provided by the network simulation for two simple scenarios. All the graphs shown in this section are directly generated by the planning tool as described in section 6.2.

7.3.1 Single Interference

Network Scenario

The sample network (figure 7.2) consists of two sources which are sending traffic to two sinks through a single switch.

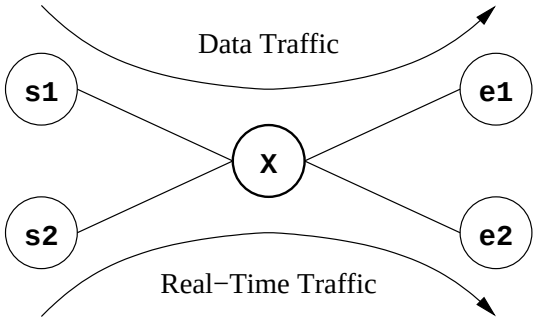


Figure 7.2: Test Network

Flow 1, representing data traffic, is modelled as bursty traffic (Markov Modulated On-Off), while flow 2, real-time traffic, is modelled using a Markov-Modulated Rate Process.

For the following two tests, the bitrate of source `s1` varies from 4 to 16 Mb/s. The sample realisation in figure 7.3 has a bitrate of 14 Mb/s. The simulation time is actually larger than the time scale shown on figure 7.3, the scale used highlights the precise behaviour of the two sources.

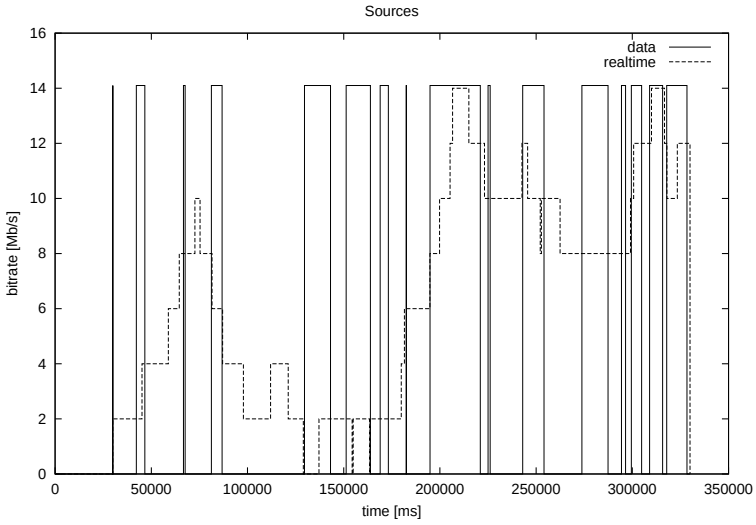


Figure 7.3: Test Sources Output

The parameters used to model these two sources have been chosen according to the measures taken on the Eircom network. However, the lack of accurate modelling for some data types has made it impossible to model some parts of the traffic mix.

Test Goal

In this experiment, the simulator tests are designed to see how one flow of real time traffic is effected when it is in competition for resources with another flow of data traffic. The Quality of Service is measured using two metrics: delay and loss ratio. Two router scheduling policies are compared: FIFO and GPS.
The following graphs (figures 7.4 and 7.5) represent the results of the simulation for various values of $b(s1)$. In the first configuration (on the left), the two sources share resources with equal priority and the switch is modelled as a single FIFO queue. In the second configuration (on the right), each flow has a part of the bandwidth reserved. It may use more bandwidth if and only if the other flow is idle.

Results: Delay

The effect on the scheduling policy of the average delay for flow 1 is shown of figure 7.4.

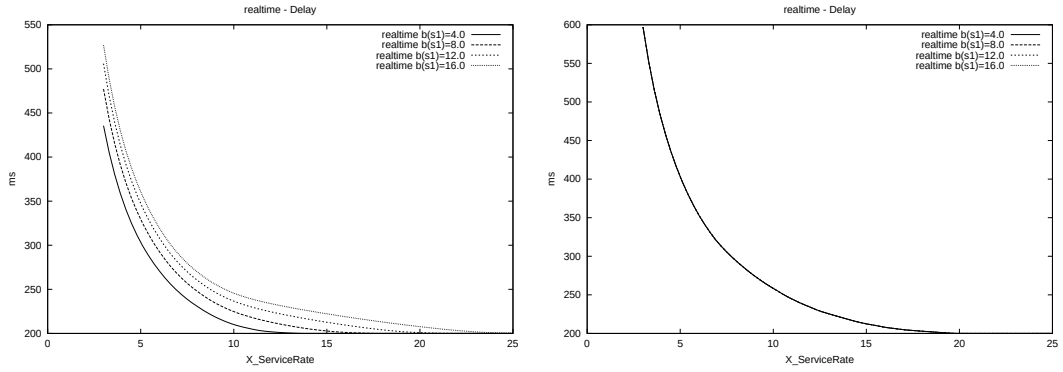


Figure 7.4: Delay / Service Rate

Results: Loss Ratio

The loss ratio effecting flow 1 is shown on figure 7.5.

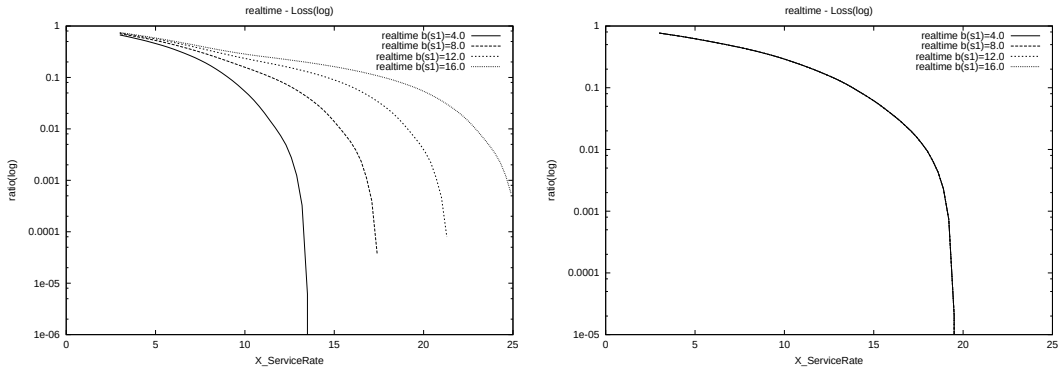


Figure 7.5: Loss / Service Rate

Analysis

Two conclusions can be drawn from this first experiment:

- *Influence of the scheduling policy*: the values of loss ratio and delay depend of the bitrate of the source `s1` for the FIFO system, whilst they are the same in the second case where the use of GPS allows the isolation of the two flows. In the second test, where the two flows are isolated, the delay and loss for the real time traffic improves only for higher values of $b(s1)$. This can be explained by the fact that in the first case, flows use bandwidth according to their respective weights. For the second case, `s2` uses idle bandwidth from `s1` only when it is idle. The idle period is not affected by the bitrate.

  Note that in both cases, the delay is converging toward a minimum value of 200ms, which is the theoretical minimum value for these scenarios (two links with a 100ms delay). The extra delay is due to the buffering of data, as the computing time inside the router is neglected.

- *Link dimensioning*: the loss ratio drops quickly for a certain value of the service rate of the switch. This value represents the bandwidth needed for the resources to be sufficient for the two flows. This can be used in network planning to determine the bandwidth needed on a given link.

Buffer size influence

For the first experiment, the buffer size was fixed at 200B. Here, using the same network scenario, we are studying the influence of the buffer size on flow 1 performance (figure 7.6).

We can distinguish two domains in these graphs:

- if the buffer size is approximately lower than 250B: the loss ratio is slowly decreasing. As the buffer size increases, more packets can be stored, thus decreasing the loss ratio.

- if the size is larger than this value: The loss ratio decreases very rapidly. This is due to the fact that the buffer can now store all the data generated during the bursts.

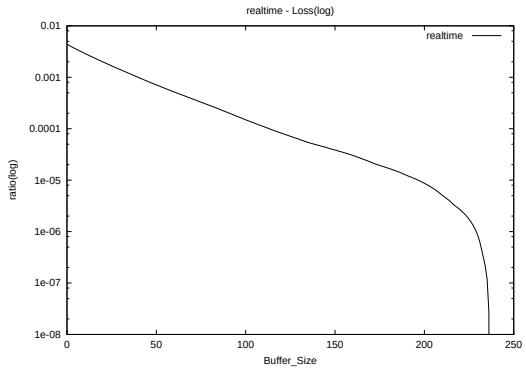


Figure 7.6: Buffer size influence

As the first experiment was based on a buffer size of 200B, the measurements provided are valid and not biased by an overly large buffer size.

7.3.2 Double Interference

Network Scenario

Here we study the influence of two intermediate flows on one main flow. The results are shown in figure 7.7. The main flow is modelled as a `realtime` flow, as in the previous example, while the interfering flows are modelled as `data`. The scheduling policy considered is FIFO. The metrics studied are the as before, namely loss and delay. This section concludes with a comparison between the respective influence of the flows A and B.

Results

The results are displayed in figure 7.8. The parameters for the first column are taken from $b(B) = 6Mb/s$, with $b(A)$ varying, while for the second column, $b(A) = 6Mb/s$, $b(B)$ varying.

The conclusions are quite similar to the observations made in the previous section. The influence on delay is not really important given the values chosen, and we can

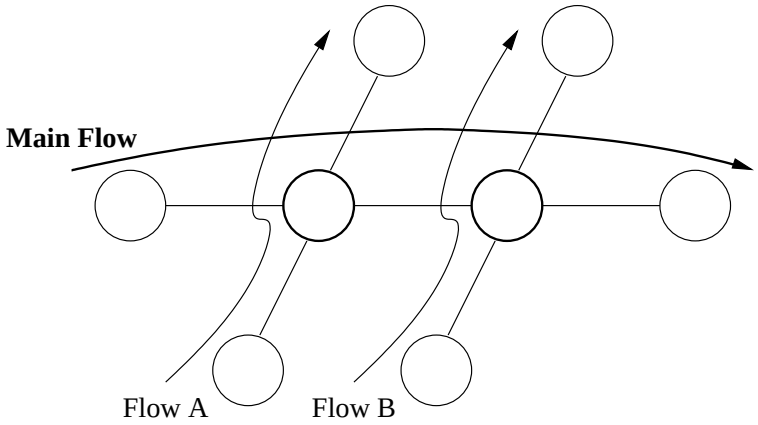


Figure 7.7: Test Network

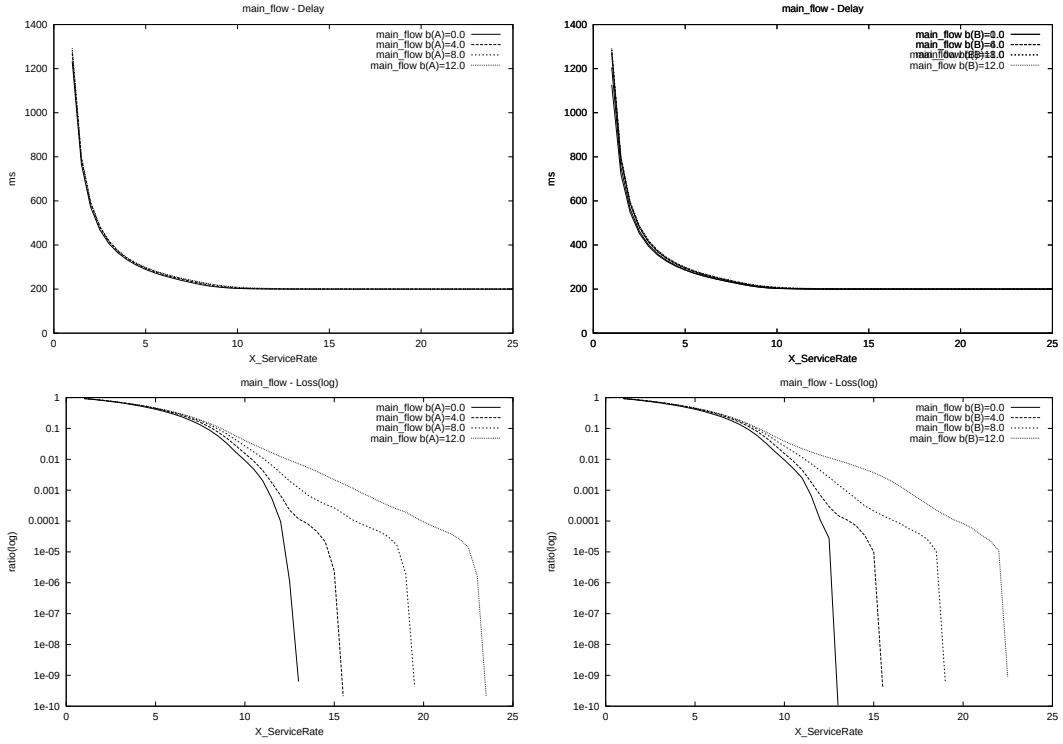


Figure 7.8: Loss Ratio Study

observe the same drop in the loss rate when the bandwidth is large enough to handle the traffic.

Respective influence of each flow

To study the respective influence of flows A and B, simulations based on the same network scenario were run, with slightly different burst bitrate values for A and B. The results are displayed on figures 7.9 and 7.10.

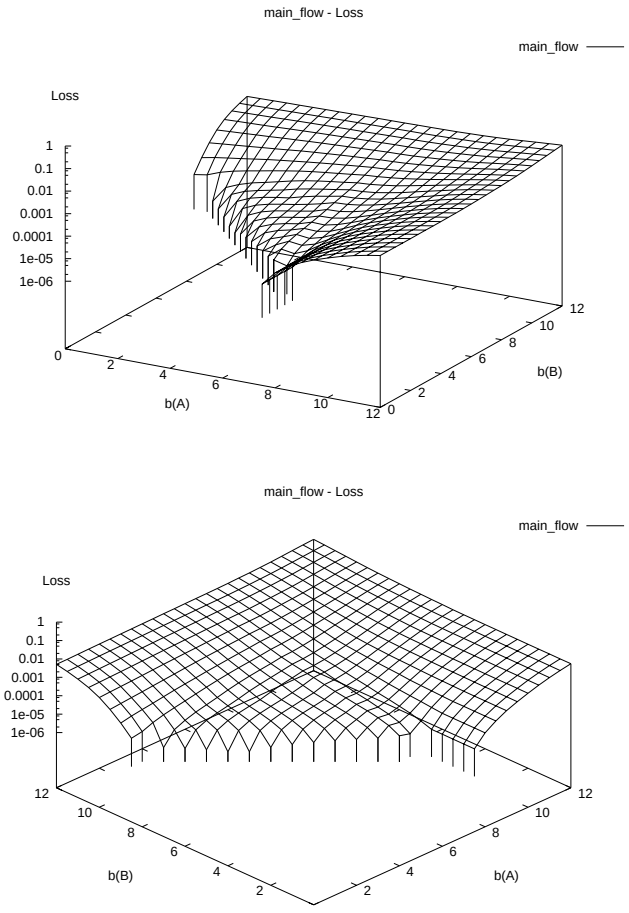


Figure 7.9: Loss Ratio Study - A/B

From figure 7.9, which shows the same data from 2 different angles of observation, we can say that the influence of A and B is roughly equivalent, as the graph looks

symmetric.

However, to determine this equivalence more precisely, the study shown in figure 7.10 is used in a separate experiment. Here we consider two cases for each flow: a low rate ($b = 2Mb/s$) and a high rate ($b = 10Mb/s$). The four possible cases are displayed on the graph. Focusing on the two plots in the middle, which corresponds to an exchange of the bit rates of flow A and B, we can actually see that the impact is similar but slightly different.
This is due to the fact that the main flow rate is actually shaped by going through the first switch, where it competes with flow A, before going through the second switch where it competes with flow B.

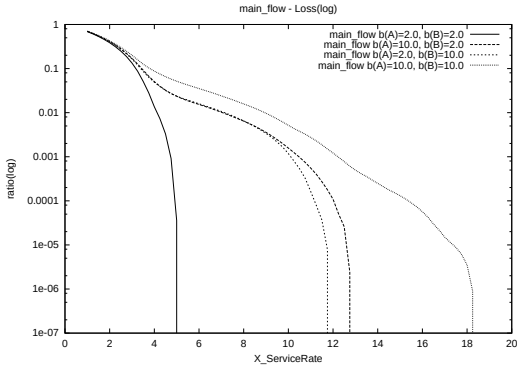


Figure 7.10: Loss Ratio Study - A/B

7.4 Conclusion

This section used the abilities of the software developed to examine some simple network scenarios. Through the experiments outlined above both bandwidth requirements and quality of service have been studied. The network simulator has been successfully used to provide results that would be of interest to telecommunications network planners.

Chapter 8

Conclusion

8.1 Objectives Fulfilled

The aim of this work was to provide a computer-based tool to analyse and forecast the behaviour of Next Generation Networks compared to switched telephony ones. The benefits and the limitation of the migration process can be estimated through analysis and fast simulation.

The reimplementation of [Hov01] has been successfully achieved. The performances and the scalability have been drastically improved. The results and analysis of [Hov01] relating to the bandwidth gain when carrying traffic over ATM instead over PSTN, have been validated. However, a deeper analysis has shown that the gain in bandwidth obtained by carrying traffic over ATM instead of PSTN may be overestimated.

Moreover, the implementation of an event-driven simulator on top of the existing infrastructure has facilitated measurements of data depending on dynamic behaviour of the traffic namely delay and loss. These are used to estimate the QoS assured on a given network. The number of available traffic models has been increased. The traffic management policies are simple, nevertheless they are representative of the policies

which are currently used on the network. The study of the dynamics of data networks is now possible using the planning tool. Assumptions were made as in [Hov01]: provide an estimation of a network behaviour using fast algorithms.

Emphasis has been put on the interface: the results format is targeted for a direct comparison. The influence of one or several parameters can be studied graphically, with only one script describing the network to be studied.

This work provides network operators with an improved method of evaluating the migration to Next Generation Networks. The bandwidth requirement evaluation is faster, and the planning tool is now capable of evaluating some QoS parameters of the planned NGN.

8.2 Obstacles Overcome

One of the main difficulties encountered was the quantity of work previously done which had to be studied and reviewed. Much work has been done on network modelling and simulation, and the selection of relevant aspects for this project proved a difficult task.

The software development of the network simulator layer of the planning tool was much longer than anticipated. The use of an existing network simulator, which was considered but rejected during this project, may have helped to provide significant results. The advantage of the current implementation is the integration of these two main features in one tool. Another advantage is that the algorithms used have been designed to allow an easy implementation of additional source models, and more complex scheduling policies based on the implemented ones.

8.3 Future Work

For large networks, the use of the scripting language tends to be difficult. A graphical interface should be used to replace the existing scripting system, or built on top of it.

More source models and traffic control policies should be added to allow a more accurate analysis. This will not require much work in implementation for bandwidth estimation (because we only need the maximum and the average bitrate).
For the network simulation, others sources can be added easily, as long as the evolution of the changes in bitrate is known. A major improvement may be the implementation of reactive flows, such as TCP/IP or ABR class of ATM.

For studying QoS service on a network, we used delay and loss rate as metrics. However, another metric which is useful to characterise QoS is jitter, which is the variation of delay across time.

As dynamic routing is a key to QoS policies, especially for IP, the tests had to be made on quite simple networks. Dynamic routing will ensure results closer to real systems performances. This has not been a drawback, because the networks studied were simple and did not include several paths for a flow.

Finally, a validation of the results against other simulators should be done when appropriate.

Bibliography

[AAHM01] Bengt Ahlgren, Anders Andersson, Olof Hagsand, and Ian Marsh. Dimensioning links for IP telephony. In *2nd IP Telephony Workshop*, April 2001.

[And00] Anders Andersson. Capacital study of statistical multiplexing for IP telephony. Technical Report T2000:03, SICS, 2000.

[BBE+99] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala. Improving simulation for network research. Technical Report 99-702b, USC Computer Science Departement, September 1999.

[BEF+00] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *IEEE Computer*, 33(5), 2000.

[Ben01] J.M. Bennett. Voice over packet reliability issues for next generation networks. In *IEEE International Conference on Communications*, volume 1, pages 142–145. ICC 2001, June 2001.

[CFW94] C. Courcoubetis, G. Fouskas, and R. Weber. On the performance of an effective bandwidth formula. *Proc. 14th Int. Teletraffic Cong.*, 1:201–212, June 1994.

[CGK02] Matthew Caesar, Dipak Ghosal, and Randy H. Katz. Resource management for IP telephony networks. In *International Workshop on QoS (IWQoS)*, May 2002.

[Cha99] Xinjie Chang. Network simulations with opnet. In *Proceedings of the 1999 Winter Simulation Conference*, December 1999.

[Dan02] Laurent Daniel. IP repousse ATM vers la sphère télécom. *Réseaux & Télécoms*, 200:56–59, July 2002.

[DGH01] Debojyoti Dutta, Ashish Goel, and John Heidemann. Faster network design with scenario pre-filtering. Technical Report ISI-Technical-Report-550, University of Southern California, November 2001.

[Don01] Michael Donohoe. Evolving the next generation network. Technical Report PR 109 NPD 01, Eircom, March 2001.

[Eri01a] Ericsson. Broadband multi-service networks: The advance to IP networks of the future. White Paper, October 2001.

[Eri01b] Ericsson. The migration story: Different highways to a multi-service network. White Paper, October 2001.

[GGT00] Yang Guo, Weibo Gong, and Donald F. Towsley. Time-stepped hybrid simulation (TSHS) for large scale networks. In *INFOCOM*, pages 441–450, 2000.

[Gro01] The Applied Technology Group. Next-gen VoIP services and applications using SIP and Java. Technical report, 2001.

[GW96] R. Gurski and C. L. Williamson. TCP over ATM: Simulation model and performance results. In *Proceedings of the 15th Annual IEEE International Phoenix Conference on Computers and Communications*, pages 328–335. IEEE, March 1996.

[HH01a] P. Huang and J. Heidemann. Capturing TCP burstiness for lightweight simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*. Society for Computer Simulation, January 2001.

[HH01b] Polly Huang and John Heidemann. Minimizing routing state for ligthweight network simulation. In *MASCOTS 2001*, August 2001.

[HKS99] Helmut Hlavacs, Gabriele Kotsis, and Christine Steinkellner. Traffic source modeling. Technical Report TR-99101, University of Vienna, 1999.

[Hov01] Eirik Hovland. Planning for migration to a next generation network. Master's thesis, Trinity Collge Dublin, Ireland, September 2001.

[HP92] Peter G. Harrison and Naresh M. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. Addison-Wesley, 1992.

[JMW96] D. Jagerman, B. Melamed, and W. Willinger. Stochastic modeling of traffic processes. *Frontiers in Queueing: Models, Methods and Problems*, 1996.

[KK01] Jung-Taek Kim and Inseon Koh. A development of an intelligent algorithm for bandwidth allocation in ATM netwoks using Petri Nets. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, October 2001.

[KS95] G. Kesidis and A. Singh. An overview of cell-level ATM network simulation. In *High Perfomance Computing Systems Conference*, 1995.

[KSCK96] G. Kesidis, A. Singh, D. Cheung, and W. Kwok. Feasibility of fluid event-driven simulation for ATM networks. In *IEEE Globecom*, 1996.

[KW93] G. Kesidis and J. Walrand. Quick simulation of ATM buffers with on-off multiclass Markov fluid sources. *ACM Tomacs*, 3(3):269–276, July 1993.

[KWC93] George Kesidis, Jean Walrand, and Cheng-Shang Chang. Effective bandwidths for multiclass markov fluids and other ATM sources. *IEEE Trans. Networking*, 1(4):424–428, August 1993.

[LFG+01] Benyuan Liu, Daniel R. Figueirdo, Yang Guo, Jim Kurose, and Don Towsley. A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation. In *INFOCOM*, 2001.

[LGK+99] B. Liu, Y. Guo, J. Kurose, D. Towsley, and W. Gong. Fluid simulation of large scale network: Issues and tradeoffs. Technical Report UM-CS-1999-038, University of Massachusetts, 1999.

[LK00] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis*. McGraw Hill, third edition, 2000.

[MBD+95] M. Ajmone Marsan, A. Bianco, T. V. Do, L. Jereb, R. Lo Cigno, and M. Munafò. ATM simulation with CLASS. *Performance Evaluation*, 24(1–2):137–159, November 1995.

[MBGT97] Marco Ajmone Marsan, K. Begain, R. Gaeta, and M. Telek. GSPN analysis of ABR in ATM LANs. In *Proceedings of the Seventh International Workshop on Petri Nets and Performance Models, June 3-6, 1997, Saint Malo, France*, pages 227–236. IEEE Computer Society, June 1997.

[MKD01] N. Milidrag, G. Kesidis, and M. Devetsikiotis. An overview of fluid-based quick simulation techniques for large packet-switched communication networks. In *Proc. SPIE ITCom*, August 2001.

[MM97] Sándor Molnár and György Miklós. On burst and correlation structure of teletraffic models. In *5th IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, July 1997.

[MM99] S. Molnár and I. Maricza. Source characterization in broadband networks. Interim report, COST 257, January 1999.

[MS81] William M. McCormack and Robert G. Sargent. Analysis of future event set algorithms for descrete event simulation. *Communications of the ACM*, 24(12), December 1981.

[Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4), April 1989.

[Net99] Nortel Networks. IP traffic engineering for carrier networks: Using constraint-based routing to deliver new services. White Paper, October 1999.

[PD00] Larry L. Peterson and Bruce S. Davie. *Computer Networks, A Systems Approach*. Morgan Kaufmann, second edition, 2000.

[PE96] Harry G. Perros and Khaled M. Elsayed. Call admission control schemes: A review. *IEEE Communications Magazine*, 34(11):82–91, 1996.

[Peu99] Markus Peuhkuri. IP quality of service. Technical Report Tik-110.551, Helsinki University of Technology, Laboratory of Telecommunications Technology, May 1999.

[RK96] A. Rueda and W. Kinsner. A survey of traffic characterization techniques in telecommunication networks. In *Proceedings of the 1996 IEEE Candian Conference on Electrical and Computer Engineering*, May 1996.

[SUOS01] Kohei Shimoto, Masanori Uga, Masaaki Omotani, and Shigeki Shimizu. Design of IP+ATM switch router architecture for best-effort and guaranteed services the next generation public data networks. In *IEEE International Conference on Communications*, volume 1, pages 89–94. ICC 2001, June 2001.

[WUX98] Carey L. Williamson, Brian W. Unger, and Zhonge Xiao. Parallel simulation of ATM networks: Case study and lessons learned. Technical Report

TeleSim, Department of Computer Science, University of Calgary, April 1998.

[YG99] Anlu Yan and Wei-Bo Gong. Time-driven fluid simulation for high-speed networks. *IEEE Transactions on Information Theory*, 45(5), July 1999.

[YMTB01] T.K. Yung, J. Martin, M. Takai, and R. Bagrodia. Integration of fluid-based analytical model with packet-level simulation for analysis of computer networks. In *SPIE*, 2001.