DYNAMIC MANAGEMENT AND RESTORATION OF VIRTUAL PATHS IN BROADBAND NETWORKS BASED ON DISTRIBUTED SOFTWARE AGENTS

Pere VILÀ TALLEDA

ISBN: 84-688-8632-7
Dipòsit legal: Gi.1116-2004
http://hdl.handle.net/10803/7726

Universitat de Girona

Department of Electronics, Computer Science and Automatic Control

PhD Thesis

Dynamic Management and Restoration of Virtual Paths in Broadband Networks based on Distributed Software Agents

Author: Pere Vilà

Supervisor: Josep Lluís Marzo

Thesis presented in fulfillment of the requirements for
the degree of PhD in Computer Engineering.
Girona, February 2004

Dr.	
President	

Dr.	
Secretari	

Dr.	
Vocal	

Dr.	
Vocal	

Dr.	
Vocal	

Data de la defensa pública	
Qualificació	

a la Teresa, per la seva paciència, suport i amor

Acknowledgements

I would like to express my sincere thanks to my thesis director Dr. Josep Lluís Marzo, for trusting in my commitment, and for his guidance and encouragement.

I am also extremely grateful to Dr. David Harle from the University of Strathclyde in Glasgow, where I began my research work under his guidance in a four-month research stay. I am also grateful to Dr John Bigham who took me into his group for three months in Queen Mary College – University of London (UdG grant, 2001). Many thanks to Dr. Franco Davoli from the University of Genoa with whom I discussed several issues during two weeks in Genoa, (Spanish Ministry of Education HI98-02).

I wish to express my gratitude to all members of the Broadband Communications and Distributed Systems Group: Anna, Liliana, David, César, Clara, Martí, Antoni, Teo, Joan, and especially to Lluís, Antonio, Eusebi, Santi and Ramon, who helped me in different circumstances.

Special thanks go to Dani Massaguer, who has collaborated with me as an undergraduate student participating in several grants and projects related with this work.

Finally, I wish to thank the many people who have, in one way or another, made this thesis possible. I apologise for not listing everyone here.

This work has been partially supported by the Ministry of Science and Technology of Spain under contracts MCYT TIC2003-05567, MCYT TIC2002-10150-E, CICYT TEL99-0976, and by the UdG research support fund (UdG-DinGruRec2003-GRCT40).

Abstract

Network management is a wide field including topics as diverse as fault restoration, network utilisation accounting, network elements configuration, security, performance monitoring, etc. This thesis focuses on resource management of broadband networks that have the mechanisms for performing resource reservation, such as Asynchronous Transfer Mode (ATM) or Multi-Protocol Label Switching (MPLS). Logical networks can be established by using Virtual Paths (VP) in ATM or Label Switched Paths (LSP) in MPLS, which we call generically Logical Paths (LP). The network users then use these LPs, which can have pre-reserved resources, to establish their communications. Moreover, LPs are very flexible and their characteristics can be dynamically changed. This work focuses, in particular, on the dynamic management of these logical paths in order to maximise the network performance by adapting the logical network to the offered connections.

In this scenario, there are several mechanisms that can affect and modify certain features of the LPs (bandwidth, route, etc.). They include load balancing mechanisms (bandwidth reallocation and re-routing) and fault restoration (utilisation of backup LPs). These two mechanisms can modify the logical network and manage the resources (bandwidth) of the physical links. Therefore, due to possible interferences, there is a need to co-ordinate these mechanisms. Conventional resource management, using a logical network, performs a centralised recalculation of the whole logical network periodically (e.g. every hour / day). This brings the problem that the logical network readjustments do not happen when a problem occurs. Moreover, there is a need of maintaining a centralised network overview. Management is becoming more difficult and complex due to increasing network sizes and speeds and new service requirements. Network administrators need more and more powerful applications to facilitate their decisions and, despite their experience, they are prone to mistakes. That is why there is a trend in the network management field towards automating and distributing the network management mechanisms.

In this thesis, a distributed architecture, based on a Multi-Agent System (MAS), is proposed. The main objective of this architecture is to perform joint resource management at a logical network level, integrating the bandwidth reallocation and LP re-routing with pre-planned restoration and spare bandwidth management. This is performed continuously, not periodically, when a problem is detected (an LP is congested, i.e. it is rejecting new user connections because it is already saturated with user connections) in a completely distributed way, i.e. without any central network overview. Therefore, the proposed architecture performs small rearrangements in the logical network and thus it is continuously being adapted to the user demands. The proposed architecture also considers other objectives, such as scalability, modularity, robustness, simplicity and flexibility.

The proposed MAS is structured in two layers of agents: The network Monitoring (M) agents and the Performance (P) agents. All these agents are situated at different network nodes, where the computing facilities are. There is one P agent and several M agents on every node. The M agents are subordinated to the P agents, therefore the proposed architecture can be seen as a hierarchy of agents. Each agent is responsible for monitoring and controlling the resources they are assigned to. Each M agent is assigned to an LP and each P agent is responsible for a node and the outgoing physical links. M agents' main tasks are the detection of congestion in the LPs and the switchover mechanism when a failure occurs. They must react quickly when an event occurs; therefore, they are pure reactive agents. Each P agent maintains a node status and keeps track of the established LPs starting at or going through that particular node. For every physical link, it maintains the status of the established LPs, the bandwidth assigned to them, the spare bandwidth reserved for the backup paths, the backup paths configuration, the available bandwidth on the link, etc. They are responsible for receiving the failure alarms from other P agents and also from the lower layer. Each P agent also maintains a partial logical network view and communicates and collaborates with other P agents.

We have performed several experiments, using a connection level distributed simulator of our own design. The results show that our architecture is capable of performing the assigned tasks of detecting congestion, dynamic bandwidth reallocation and re-routing in a co-ordinated way with the pre-planned restoration and the spare capacity management. The distributed architecture offers a suitable scalability and robustness due to its flexibility and modularity.

Resum

La gestió de xarxes és un camp molt ampli i inclou aspectes com ara la restauració de fallades, la comptabilitat de l'ús de la xarxa, la configuració dels elements de la xarxa, la seguretat, la monitorització del rendiment, etc. Aquesta tesi doctoral està centrada en la gestió dels recursos en les xarxes de banda ampla que disposin de mecanismes per fer reserves de recursos, com per exemple *Asynchronous Transfer Mode* (ATM) o *Multi-Protocol Label Switching* (MPLS). Es poden establir xarxes lògiques utilitzant els *Virtual Paths* (VP) d'ATM o els *Label Switched Paths* (LSP) de MPLS, als que anomenem genèricament camins lògics. Els usuaris de la xarxa utilitzen doncs aquests camins lògics, que poden tenir recursos assignats, per establir les seves comunicacions. A més, els camins lògics són molt flexibles i les seves característiques es poden canviar dinàmicament. Aquest treball, se centra, en particular, en la gestió dinàmica d'aquesta xarxa lògica per tal de maximitzar-ne el rendiment i adaptar-la a les connexions ofertes.

En aquest escenari, hi ha diversos mecanismes que poden afectar i modificar les característiques dels camins lògics (ample de banda, ruta, etc.). Aquests mecanismes inclouen els de balanceig de la càrrega (reassignació d'ample de banda i reencaminament) i els de restauració de fallades (ús de camins lògics de *backup*). Aquests dos mecanismes poden modificar la xarxa lògica i gestionar els recursos (ample de banda) dels enllaços físics. Per tant, existeix la necessitat de coordinar aquests mecanismes per evitar possibles interferències. La gestió de recursos convencional que fa ús de la xarxa lògica, recalcula periòdicament (per exemple cada hora o cada dia) tota la xarxa lògica d'una forma centralitzada. Això introdueix el problema que els reajustaments de la xarxa lògica no es realitzen en el moment en què realment hi ha problemes. D'altra banda també introdueix la necessitat de mantenir una visió centralitzada de tota la xarxa. La gestió de xarxes s'està fent cada vegada més difícil i complexa degut a la creixent mida i velocitat i a la introducció de nous requeriments de servei. Això pot fer que malgrat la seva experiència els administradors de xarxa puguin cometre equivocacions, a més necessiten cada vegada aplicacions més potents per facilitar la seva tasca. És per tots aquests fets que hi ha la tendència, en el camp de la gestió de xarxes, cap a una automatització i distribució dels mecanismes de gestió de xarxa.

En aquesta tesi, es proposa una arquitectura distribuïda basada en un sistema multi agent. L'objectiu principal d'aquesta arquitectura és realitzar de forma conjunta i coordinada la gestió de recursos a nivell de xarxa lògica, integrant els mecanismes de reajustament d'ample de banda amb els mecanismes de restauració preplanejada, inclosa la gestió de l'ample de banda reservada per a la restauració. Es proposa que aquesta gestió es porti a terme d'una forma contínua, no periòdica, actuant quan es detecta el problema (quan un camí lògic està congestionat, o sigui, quan està rebutjant peticions de connexió dels usuaris perquè està saturat) i d'una forma completament distribuïda, o sigui, sense mantenir una visió global de la xarxa. Així doncs, l'arquitectura proposada realitza petits rearranjaments a la xarxa lògica adaptant-la d'una forma contínua a la demanda dels usuaris. L'arquitectura proposada també té en consideració altres objectius com l'escalabilitat, la modularitat, la robustesa, la flexibilitat i la simplicitat.

El sistema multi agent proposat està estructurat en dues capes d'agents: els agents de monitorització (M) i els de rendiment (P). Aquests agents estan situats en els diferents nodes de la xarxa: hi ha un agent P i diversos agents M a cada node; aquests últims subordinats als P. Per tant l'arquitectura proposada es pot veure com una jerarquia d'agents. Cada agent és responsable de monitoritzar i controlar els recursos als que està assignat. Cada agent M s'encarrega d'un camí lògic i cada agent P s'encarrega d'un node i dels seus corresponents enllaços físics de sortida. La principal tasca dels agents M és detectar la congestió en els camins lògics i activar el mecanisme de *switchover* quan es produeix una fallada. Quan hi ha algun esdeveniment que els afecta, els agents M han de reaccionar ràpid, per tant es tracta d'agents reactius. Els agents P mantenen un estat del node en general i dels camins lògics que comencen o passen pel node en qüestió. Concretament, per cada enllaç físic es manté una llista dels camins lògics que hi passen, de l'ample de banda assignat a cadascun d'ells, de l'ample de banda reservat pels camins de *backup*, la configuració d'aquests camins de *backup*, l'ample de banda disponible a l'enllaç físic, etc. Els agents P són responsables de rebre les alarmes de fallada del mateix node (capes inferiors) o d'altres agents P. Cada agent P també manté una visió parcial de la xarxa lògica i es comunica i col·labora amb altres agents P.

S'han realitzat diferents experiments utilitzant un simulador distribuït a nivell de connexió proposat per nosaltres mateixos. Els resultats mostren que l'arquitectura proposada és capaç de realitzar les tasques assignades de detecció de la congestió, reassignació dinàmica d'ample de banda i reencaminament d'una forma coordinada amb els mecanismes de restauració preplanejada i gestió de l'ample de banda reservat per la restauració. L'arquitectura distribuïda ofereix una escalabilitat i robustesa acceptables gràcies a la seva flexibilitat i modularitat.

Table of Contents

List of Figures

List of Tables

Glossary

AAL	ATM Adaptation Layer.
ABR	Available Bit Rate.
ACL	Agent Communication Language.
ADM	Add-Drop Multiplexer.
AI	Artificial Intelligence.
ALP	Any Logical Path.
API	Application Programming Interface.
ATM	Asynchronous Transfer Mode.
BDI	Beliefs Desires Intention.
BDI	Beliefs Desires and Intentions.
BGP	Border Gateway Protocol.
B-ISDN	Broadband Integrated Service Digital Network.
CAC	Connection Admission Control.
CBP	Connection Blocking Probability.
CBR	Constant Bit Rate.
CBR	Case Based Reasoning.
CCITT	Comité Consultative Internationale de Telegraphique et Telephonique.
CDV	Cell Delay Variation.
CLP	Cell Loss Priority.
CLR	Cell Loss Rate.
CMIP	Common Management Information Protocol.
CORBA	Common Object Request Broker Architecture.
CoS	Class of Service.
CRC	Cyclic Redundancy Check.
CR-LDP	Constraint-based Routing LDP.
CTD	Cell Transfer Delay.
DAI	Distributed Artificial Intelligence.
DiffServ	Differentiated Services.
DME	Distributed Management Environment.
DNS	Domain Name Service.
DWDM	Dense Wavelength Division Multiplexing .

FBO	Free Bandwidth Only.
FCAPS	Fault, Configuration, Accounting, Performance and Security.
FEC	Forwarding Equivalent Class.
FIPA	Foundation for Intelligent Physical Agents.
FNO	First Node Only.
FR	Frame Relay.
FSC	Fibre Switch Capable.
GFC	Generic Flow Control.
GMPLS	Generalised Multiprotocol Label Switching.
GoS	Grade of Service.
HEC	Header Error Check.
IETF	Internet Engineering Task Force.
IGP	Interior Gateway Protocol.
ILM	Intermediate Level Manager.
IP	Internet Protocol.
IS-IS	Intermediate System to Intermediate System.
ISO	International Organisation for Standardisation.
ISP	Internet Service Providers.
ITU-T	International Telecommunication Union – Telecommunication Standardisation Section.
JDMK	Java Dynamic Management Kit.
KQML	Knowledge Query and Manipulation Language.
L2SC	Layer-2 Switch Capable.
LAN	Local Area Network.
LDP	Label Distribution Protocol.
LM	Layer Manager.
LMP	Link Management Protocol.
LP	Logical Path.
LSC	Lambda Switch Capable.
LSI	Label Stack Indicator.
LSP	Label Switched Path.
LSR	Label Switched Routers.
MAC	Medium Access Control.
MAC	Medium Access Control.
MAS	Multi-Agent System.
MbD	Management by Delegation.
MIB	Management Information Base.

MO	Managed Object.
MPLS	Multiprotocol Label Switching.
NCC	Network Control Centre.
NE	Node Emulator.
NME	Network Management Entity.
NNI	Network-Network Interface.
OAM	Operations and Maintenance.
OC	Offered Connections.
OMG	Open Management Group.
OS	Operating System.
OSF	Open Software Foundation.
OSI	Open Systems Interconnection.
OSPF	Open Shortest Path First.
OXC	Optical Cross-Connects.
PDU	Protocol Data Unit.
PPP	Point to Point Protocol.
PSC	Packet Switch Capable.
PT	Payload Type.
PXC	Photonic Cross-Connect.
QoS	Quality of Service.
RC	Rejected Connections.
RMI	Remote Method Invocation.
RMI	Remote Method Invocation.
RMON	Remote Monitoring.
RSVP	Resource Reservation Protocol.
SDH	Synchronous Digital Hierarchy.
SDH	Synchronous Digital Hierarchy.
SLA	Service Level Agreements.
SMO	Systems Management Overview.
SNMP	Simple Network Management Protocol.
SONET	Synchronous Optical Network.
SONET	Synchronous Optical NETwork.
TCP	Transport Control Protocol.
TDM	Time-Division Multiplexing.
TE	Traffic Engineering.
TEG	Traffic Event Generator.
TF	Triggering Function.

TLM	Top Level Manager.
TMN	Telecommunication Management Network.
TTL	Time To Live.
UBR	Unspecified Bit Rate.
UDP	User Datagram Protocol.
UNI	User-Network Interface.
UPC	Usage Parameter Control.
URL	Uniform Resource Locator.
VBR	Variable Bit Rate.
VC	Virtual Channel.
VCI	Virtual Channel Identifier.
VP	Virtual Path.
VPI	Virtual Path Identifier.
VPN	Virtual Private Networks.

Chapter 1 Introduction

1.1 Overview

This chapter describes the motivation for this work in reference to certain problems and new trends we have detected in the field of network management. From this starting point, the main thesis objectives are pointed out, along with the expected achievements. The chapter ends by describing the structure and contents of the document.

1.2 Motivation

Network management deals with maintaining correct operations in network infrastructures. A typical description of network management functions would include mechanisms for avoiding network failures and restoring them as fast as possible when they occur; maintaining the correct configuration of the equipment and software; accounting the network utilisation (for several reasons, e.g. detecting problems, charging users, etc); maintaining a high network performance, i.e. trying to take on as much traffic as possible using the same network resources without service degradation; and other similar functions. Hence, network management is a broad field that encompasses a large number of mechanisms. It also has a close relationship with network control (automatic mechanisms closely related to hardware, e.g., packet scheduling and buffer management) because network management typically configures and controls them from a higher level.

The network administrator typically has one or more powerful user interface with which to manage the network. This defines a cycle of monitoring the network operation and carrying out actions on it. Originally, network management involved manual reconfiguration of the network elements, but the mechanisms have since evolved into a complete set of powerful tools that help the network administrator in the planning and configuration of the network. In the beginning, these mechanisms were concentrated at the administrator's computer, mainly because of the lack of processing power at the network elements. This processing power has been increased over time and nowadays, management mechanisms tend to be spread over the network. This increasing processing power in the network elements is one of the basic reasons for the trend towards automating many network management functions that are currently performed manually by the network administrator.

We detected several other factors that have had an influence on this recent trend of automating and distributing the management mechanisms over the network:

- Management complexity: Management has become more difficult and complex due to the increasing network sizes and speeds, along with the new service requirements. Network administrators need more and more powerful applications to help them with their decisions and, despite their experience, they are prone to mistakes.
- Centralised architecture bottleneck: There are several management mechanisms that should still be performed in a centralised way; however, more and more mechanisms tend to be distributed because a bottleneck appears when this is performed in a centralised way due to the increasing number of parameters to monitor and control. There can be various types of bottleneck: a processing bottleneck due to too much data to process, a time bottleneck i.e. when the time taken to make a decision and apply that decision to the network elements, is too long, etc.
- New network technologies and services: Network technologies are becoming faster and faster and require faster response times from the network management cycle. These new technologies and services also introduce the necessity of using new management mechanisms in addition of the previous ones.
- Increasing competition: Network Providers want to get the maximum profit out of their networks in an increasingly competitive market. In order to offer lower prices, they must achieve maximum performance from their network resources.
- Difficulty of traffic prediction: new services and applications (peer-to-peer, multimedia, networked games, etc.) may cause the traffic patterns to present fluctuations which make it difficult to forecast traffic load. Therefore it is difficult to apply pre-established, hourly or daily network configurations.

Current network management systems, mostly based on management standards, have become huge applications – highly complex and difficult to maintain. Therefore, these management systems, after their design and implementation, typically become very rigid and difficult to update and upgrade with new management mechanisms. We have detected not only a need for flexibility, but also to shorten the life cycle of management systems.

There have been many attempts to perform network management using Software Agents. There are two main areas that have focussed on network management: Mobile Agents and Multi-Agent Systems (MAS). Mobile agents are a good solution in certain specific cases where the mobility brings a true benefit over other possibilities. For instance, the lower the network throughput or availability, or the higher its latency or cost, the greater is the advantage of

Mobile Agents [Bohoris et al., 2000]. Therefore, the MAS are also a good solution in many cases, i.e. homogeneous core networks with a high availability, which is the scenario we focus on. There have been several MAS proposals in this area. However, most of them fail in the definition of highly complex scenarios with many different types of agents and many different complex interactions. This brings us again to a very complex system design and maintenance where the modification and introduction of management mechanisms are both difficult.

On analysing this situation, we found that, on the one hand, there is the need to automate and distribute management mechanisms. On the other hand, there is the problem that the management mechanisms are hugely complex systems. Comparing this situation with other kinds of complex systems, e.g. operating systems, we can see that there is a core module or kernel and many other modules which are more or less independent from each other. This modular architecture brings in three main advantages: the possibility of using several modules or none, depending on the scenario and thus adapting to it; the easy modification, substitution and upgrading of these modules; and the possibility of having different versions of the same module adapted for different situations.

Therefore, we believe that in the network management environment, it could be possible to apply a similar modular architecture. The hypothetical scenario could be a "network kernel system" which could be utilised in many different types of networks and a set of modules, where each one performs one or a reduced group of management functions. These modules should be as independent as possible from the others and each one could utilise a different management architecture.

Given the experience of our research group in core networks and network technologies such as Asynchronous Transfer Mode (ATM) and Multi-Protocol Label Switching (MPLS), along with the perceived need for achieving a better network performance for the network providers, we propose an architecture for a network management module in a connection-oriented network scenario with resource reservation mechanisms. This module will be devoted to network resource management using the concept of logical or virtual network, i.e. composed of logical or virtual paths. There are other mechanisms that also make use of the logical network properties. Among these techniques, there are the fault management mechanisms based on the establishment of backup paths, i.e. pre-planned restoration mechanisms. There is also a need to analyse the relationship and the possible interference between the network resource management and the fault management techniques, both of which act over the logical network.

Finally, our proposed architecture must be evaluated. It is difficult to determine how well a management mechanism works, because comparisons are extremely difficult. Management architectures and systems are hugely complex systems and most of the approaches do not give details of how they are implemented in a way that means they can be reproduced. Therefore, the architecture must be evaluated by utilising and/or defining the appropriate metrics and comparing different configurations. Of course, it is impossible to have access to a real core network in order to test the system, this is the reason for validating the proposed architecture by means of simulations.

1.3 Objectives

Based on the motivations mentioned above, the main objective of this thesis is the proposal for a joint network resource and fault management architecture based on MAS. This architecture should be able to perform these network resource management functions at a logical network level in a completely distributed way. This requires not only the definition of the different agents, how they interact and where they are placed, but also the desired properties and characteristics the architecture should accomplish. These properties include achieving good scalability, modularity, robustness, simplicity and flexibility. It is also important, prior to evaluating the proposed architecture, to study how to adapt the resource and fault management mechanisms to it. This requires the proposal of new versions of these mechanisms that could be used by the agents, i.e. new algorithms and protocols.

Other important objectives of this thesis include firstly a background study which, in this case, is very wide-ranging, comprising network management architectures, core network technologies with reservation mechanisms and MAS applied to network management and more specifically to network resource management. With regard to the management architectures, we first study the management standards, (basically the Open Systems Interconnection –OSI– standards, the Telecommunication Management Network (TMN) standards and the Simple Network Management Protocol (SNMP), which is the Internet management standard and the most used). We focus on the identification and characterisation of the different management architectures. The studied network technologies basically comprise ATM and MPLS, with an incursion into Generalised MPLS (GMPLS). There are too many MAS devoted to network and service management. The objective was to focus on the MAS that act in a similar scenario and/or perform similar functions to our proposed system. Although we do not propose the use of Mobile Agents, it is also important to give a review of proposals based on them.

Secondly, we need to study the management techniques which could be utilised by the proposed architecture, including bandwidth reallocation, fault restoration mechanisms and spare bandwidth management. It is necessary to understand clearly how these techniques work and the possible interrelation and/or interference between them. It is also necessary to define the desired characteristics and properties of the proposed architecture.

Finally, in order to evaluate our proposed architecture, we have to perform the necessary simulations and tests. These must determine whether the architecture accomplishes the required properties while achieving the desired resource and fault management functions.

1.4 Outline of the Thesis

This document is organised into 6 chapters with a bibliography section at the end.

Chapter 2 is clearly divided into three main parts. In the first, we review the network management functions and standards, give a classification of the management architectures and their characteristics and describe recent trends in network management. In the second part, we summarise the background to network technologies, which have reservation mechanisms that allow dynamically manageable logical networks to be established. In the last section, we introduce Software Agents and study different proposals in the field of network management, presenting some examples and giving references to other articles on the state of the art in this area.

In Chapter 3, we present the network resource management mechanisms, more specifically the bandwidth reallocation and logical path re-routing. We also introduce the pre-planned restoration mechanism based on backup paths, along with the technique of bandwidth sharing among several backup paths, according to the desired protection level. This chapter also includes the desired properties and characteristics and the objectives of the proposed architecture.

In Chapter 4, we present our proposed architecture based on MAS and it is clearly divided in two main parts. In the first, we give details of the proposed architecture by presenting the different types of agents, their objectives, their interactions, the agents' distribution, etc. In the second part, we present the several network resource and fault management algorithms adapted to the MAS, along with the decision making mechanisms utilised in the agents. More specifically, we also propose a simple mechanism in order to decide whether or not a logical path is congested and the co-ordination constraints between the bandwidth management and the fault restoration mechanisms.

Chapter 5 includes the analysis and evaluation of the proposed architecture. We present several results to demonstrate the achievement of the objectives and the correct operation of the different mechanisms. We also evaluate how the system as a whole performs, by means of simulation results in different scenarios.

Finally, in Chapter 6, we conclude this document and summarise the main contributions. We also list the future work.

Chapter 2 Background

2.1 Overview

This chapter presents a brief background to the context in which this work is located. Basically, the main framework is network management and control, focusing on network resource management. Therefore, this chapter introduces the network management basics and the network technologies we have considered, mainly ATM and MPLS. We also briefly describe the GMPLS, which considers optical networks. The network resource management is then described in chapter 3, due to its relevance to the proposal of this thesis. Finally, we describe the use of Software Agents in the telecommunications world, giving an initial broad classification and some examples of related works.

2.2 Network Management

Network management and network control deal with all the functions that make a network work properly. There exist many standards in this field and this section does not present an exhaustive analysis, but the basic ideas of the main approaches and references for further background reading. A more detailed analysis can be found for instance in [Sloman, 1996], [Black, 1994], [Raman, 1998] and the first part of [Pras, 1995].

Traditionally, network management functions have been classified in many ways. This section presents two different classifications, one grouping the functions in five management functional areas, along with an overview of the management standards. The second classification is done in terms of the distribution of the decision making and the time of action of the different functions.

2.2.1 Network Management Functions and Standards

The International Organisation for Standardisation (ISO) defined, as part of its OSI set of standards, the OSI Management Framework [ISO 7498-4, 1989] [Yemini, 1993]. It defines the network management problem areas, which are called the five functional areas of OSI management. To denote these areas the term FCAPS is usually used, corresponding to the initial letters of the following five functional areas (Fault, Configuration, Accounting, Performance and Security):

Fault Management: its main task is to enable detection, isolation and correction of abnormal operation in the network. A fault is an abnormal condition that requires management attention (or action) to repair. When a fault occurs it is important, as rapidly as possible, to:

- Determine exactly where the fault is.
- Isolate the rest of the network from the failure so that it can continue to function without interference.
- Reconfigure or modify the network in such a way as to minimise the impact on operations without the failed component or components.
- Repair or replace the failed components to restore the network to its initial state.

When a fault occurs, the user generally expects to receive immediate notification and expects that the problem will be corrected almost immediately. To provide this level of fault resolution requires very rapid and reliable fault detection and diagnostic management functions. The impact and duration of faults can be minimised by the use of redundant components and alternate communication routes, to give the network a degree of fault tolerance. Fault management itself should be redundant to increase network reliability.

Configuration Management: these are the facilities that exercise control over, identify, collect data from and provide data to managed objects. Configuration management is concerned with initialising and shutting down part of the network or the entire network. It is also concerned with maintaining, adding and updating the relationships among components and the status of these components during network operation. Reconfiguration of a network is often desired in response to performance evaluation or in support of network upgrade, fault recovery or security checks.

Accounting Management: this enables charges to be made for the use of the objects managed and costs this use. Furthermore, even if no such internal charging in employed, the network manager needs to track the use of network resources by user, or user class, including user abuses of their access privileges or an inefficient use of the network. The network manager is in a better position to plan for network growth if user activity is known in sufficient detail.

Performance Management: this evaluates the behaviour of the managed objects and the effectiveness of communication activities. In some cases, it is critical to the effectiveness of an application that the communication over the network be within certain performance limits.

Performance management of a computer network comprises two broad functional categories: monitoring and controlling. Monitoring is the function that tracks activities on the network. The controlling function enables performance management to make adjustments to improve network performance. Some of the performance issues that the network manager monitors are: the level of capacity utilisation, the throughput, detecting bottlenecks and keeping response times short.

Network managers need performance statistics to help them plan, manage and maintain large networks. Performance statistics can be used to recognise potential bottlenecks before they cause problems to the end users. In this case, appropriate corrective action can then be taken. This action can take the form of changing routing tables to balance or redistribute traffic load during times of peak use or when a bottleneck is identified by a rapidly growing load in an area.

Security Management: addresses those aspects relating to security, essential to operating network management correctly and protecting managed objects. It is concerned with generating, distributing and storing encryption keys. Passwords and other authorisation or access control information must be maintained and distributed. Security management is also concerned with monitoring and controlling access to computer networks, as well as accessing all or part of the network management information obtained from the network nodes.

After the initial definition of the OSI Management Framework, the International Telecommunication Union – Telecommunication Standardisation Section (ITU-T) presented the Telecommunications Management Network (TMN) recommendations [CCITT Rec. M3010, 1992] [Sidor, 1998]. ISO also presented its OSI Systems Management Overview (SMO) [ISO 10040, 1992]. TMN includes OSI management ideas and it is possible to see TMN and OSI management as complementary to each other.

In parallel to this standardisation effort, the Internet Engineering Task Force (IETF) defined an ad hoc management protocol called Simple Network Management Protocol (SNMP) [RFC 1157, 1990]. Due to the lack of OSI and TMN based applications, manufacturers started the production of SNMP compliant systems and soon it became the de facto standard.

Typically, a network management system is a collection of tools for network monitoring and control, which is integrated into a single operator interface with a powerful but user-friendly set of commands for performing most or all network management tasks. Usually the system architecture follows the client/server model.

Initial **OSI management standards** and SNMPv1 proposed centralised management architectures only. Centralised management is based on two types of applications: The Network Management Entity (NME) or Agent (usually the server part) and the Network Control Centre (NCC) or Manager (usually the client part). Figure 2.1 shows this architecture. Note the term "Agent" in this context of network management is different from the same term in a Software Agents or Multi Agent System environment. Each network node or managed element contains a NME or Agent, which collects statistics on communications and network-related activities, stores statistics locally and responds to commands from the NCC. Possible commands include transmitting collected statistics to the NCC, changing a parameter, providing status information and generating artificial traffic to perform a test. At least one host in the network is designed as the NCC or Manager (Figure 2.2). The NCC contains management applications for data analysis, fault recovery, etc., an interface for human network managers and a database with information extracted from the Management Information Bases (MIB) from all the NMEs. NCC-NME communication is carried out using an application-level network management protocol that employs the communications architecture in the same way as any other distributed application, since management applications typically reside on the application layer. Examples of these communication protocols are SNMP and Common Management Information Protocol (CMIP) [ISO 9596, 1991]. When network elements are unable to run an Agent application or unable to communicate using management protocols, proxy systems are used.

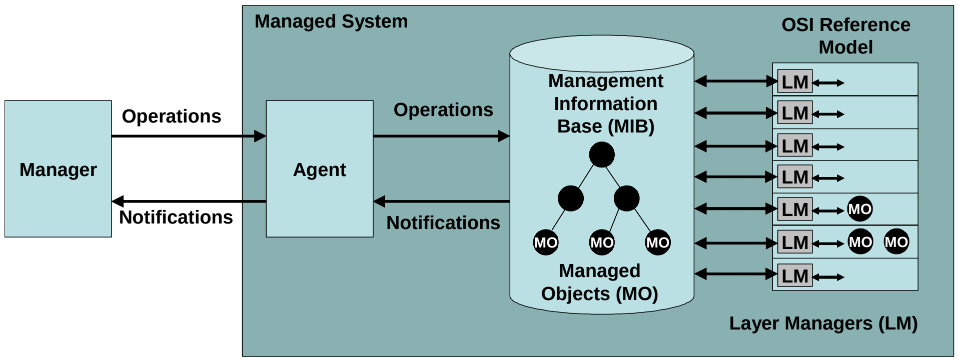


Fig. 2.1: Basic Management Model.

The management information is stored in a database called MIB on each Agent. The MIB information is an abstraction of the real managed objects that reside on the various layers of the OSI reference model of the Managed System. Layer Managers (LM) are responsible for maintaining the association between MIB information and Managed Objects (MO).

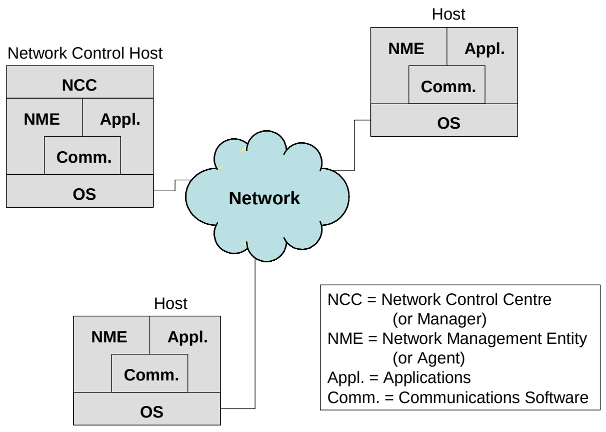

Network Control Host
NCC
NME
Appl.
Comm.
OS
Host
NME
Appl.
Comm.
OS
Network
Host
NME
Appl.
Comm.
OS
NCC = Network Control Centre (or Manager)
NME = Network Management Entity (or Agent)
Appl. = Applications
Comm. = Communications Software

Fig. 2.2: Centralised Network Management.

The main disadvantages of centralised management are low scalability and lack of robustness in the case of NCC failure/isolation. In order to minimise robustness problems, replications of the NCC were used; soon, there appeared TMN and SNMPv2, which proposed hierarchical management architectures in order to alleviate the scalability problems [Stallings, 1998a] [Stallings, 1998b]. The Remote Monitoring (RMON) [Stallings, 1998b] standard also appeared, which is used to monitor entire networks, usually Local Area Networks (LAN).

In hierarchical management, there is the concept of Manager of Managers, which is an upper layer Manager. The hierarchical management does, in fact, reduce the scalability problem, but it increases the delays and introduces more complexity.

TMN recommendations [CCITT Rec. M3010, 1992] [Sidor, 1998] proposes the use of a management network, the TMN, independent from the managed network. This separation has several advantages, e.g. better fault management capabilities, but introduces additional equipment and complexity. Moreover, the management network has also to be managed.

The TMN interfaces the managed telecommunications network at several different points. TMN recommendations also define several architectures at different levels of abstraction. TMN Information Architecture is generally based on OSI standards. At a lower level, the Physical Architecture defines how the components of the higher architectures should be mapped upon physical equipment and interfaces. TMN Functional Architecture defines Function Blocks, which contain functional components and reference points, which interconnect Function Blocks. Functional components include management application functions, MIBs, information conversion functions, human machine adaptation, presentation functions and message communication functions.

A new aspect of TMN is that it provides a structure for the multiple levels of management responsibility that already exist in real networks, known as the Responsibility Model. This brings the advantage that it becomes easier to understand and it distinguishes the various management responsibilities. The following layers are defined:

- Business management layer.
- Service management layer.
- Network management layer.
- Network element management layer.
- Network element layer.

Upper layers are more generic in functionality while lower layers are more specific. This implies a clustering of management functions into layers. In each layer, a particular management activity is broken down into a series of nested functional domains. All the interactions between domains are carried out by standardised interfaces. This provides a way of hiding objects in a subordinate domain at the next layer down. Thus, through recursion, the idea is to manage the real resources.

The logical layered architecture defines a TMN as being the overall management domain of a network provider. It also provides a logical view of the management components and their control authority, which can be put together to create the management solution.

It is worth noting here that the TMN standards are specifications and recommendations, they are not implementations. There are a number of different implementations of parts of the TMN specifications using a variety of techniques and tools. These different implementations can pass TMN compatibility tests to ensure that a specific implementation is TMN-compliant.

The **SNMP** is the most widely used protocol for the management of IP-based networks. This protocol was designed to easily provide a low-overhead foundation for multi-vendor network management of routers, servers, workstations and other network resources [Stallings, 1998a][Stallings, 1998b].

SNMP standards do not define an internet management architecture, only protocols and MIBs have been standardised. Therefore, they only define how the management information should be exchanged and what management information is provided. Agent functions can be deduced from the many MIB standards, but SNMP does not define Manager specific functions.

In SNMPv1 a single Manager may control many Agents, hence a centralised architecture is proposed. The SNMP protocol is built upon the User Datagram Protocol (UDP), which is a connectionless transport protocol (Figure 2.3). This was specified this way with the purpose of giving robustness to the management applications. Therefore, in the case of network failures, it should still be possible to exchange some parts of the management information, whereas using a connection-oriented service would just imply a connection release and nothing would be delivered. Moreover, SNMP protocol does not itself perform retransmissions and this responsibility is left to the Manager. These decisions imply that the Managers should perform a kind of polling to detect whether Agents are still operational.

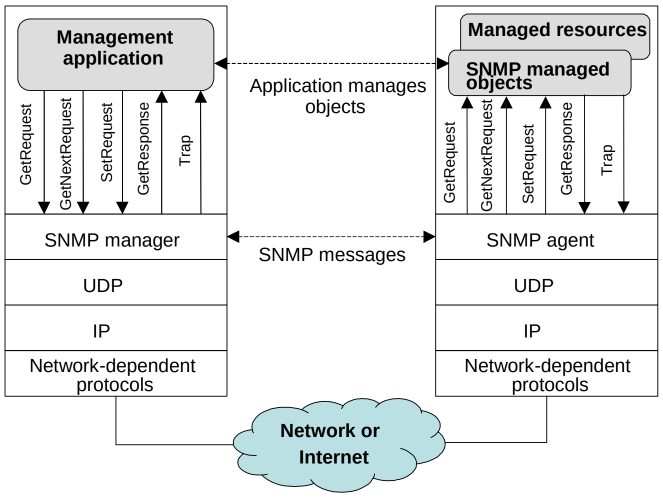


Fig. 2.3: SNMP Management.

Communication from the SNMP Manager to the SNMP Agent system is performed in a confirmed way. The Manager can take the initiative by sending three messages: *GetRequest*, *GetNextRequest* and *SetRequest*. The first two are used to retrieve management information from the Agent and the last to store or change management information. Upon receiving one of these messages, the Agent always replies with a *Response* message with the requested information or a failure indication. Finally, in the case where the Agent detects an extraordinary event, it sends an unconfirmed *Trap* message to the Manager.

SNMPv2 [Stallings, 1998a] [Stallings, 1998b] improved the SNMPv1 performance by the addition of a new message (*GetBulk*), improved its security and gave it the possibility of building a hierarchy of Managers. Regarding this last point, experience shows that it was very hard for Managers to poll hundreds of Agents. To solve this problem, Intermediate Level Managers (ILM) were introduced (Figure 2.4). Polling is performed by a number of such ILMs under control of the Top Level Manager (TLM). If an ILM detects a particular event about which the TLM wanted to be informed, it sends a special *Inform* message. At the reception of this message, the TLM directly operates upon the agent that caused the event. The main proposals in SNMPv3 are security improvements [Stallings, 1998c].

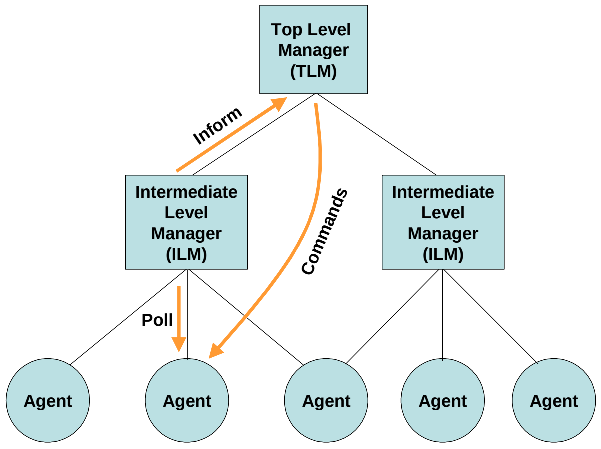


Fig. 2.4: SNMP v2 Management Hierarchy.

Network management systems are by definition distributed systems and there are also organisations for standardisation of distributed systems and many companies that propose suitable architectures for network management – both open and proprietary. Particular examples include the Distributed Management Environment (DME) from the Open Software Foundation (OSF) [OSF, 1992] [Sloman, 1996], Common Object Request Broker Architecture (CORBA) from the Open Management Group (OMG) [CORBA URL] and Java Dynamic Management Kit (JDMK) from Sun [JDMK URL]. Most of these proposals deal with generic distributed systems; for instance, DME proposes the integration of network management and service management.

2.2.2 Network Management Architectures

Network management systems are necessarily distributed, but they are special kinds of distributed systems due to the characteristics and specific problems of network management. Moreover, management systems usually take action through the use of automated functions as well as through human network manager operation or supervision. Also, it has to be taken into account that there are management functions that must be performed very quickly while others can be slower. This section presents a brief review of network management architectures, from the point of view of distributed functions and the distribution of the decision-making among the different system components. As a main rule, it is clear that the faster the management function needs to be, the closer it must be to the managed elements.

With regard to the time of action of the different network management functions, they can be classified in three groups: short term, mid term and long term; with imprecise delimitations among them. Some functions are clearly in one group whereas others could fit into different groups. It must be clarified that short term means “as fast as possible” but does not mean “real-time” or “nearly-real-time”. In this sense, there is a clear distinction between network management functions and network control functions (e.g. signalling, packet scheduling, buffer management, etc) which are beyond the scope of this work. Network control functions are always distributed. In fact, network management functions include the mid- to long-term performance monitoring and configuration of these real-time control functions. On the other hand, long term functions are usually performed by powerful analysis applications that help human network managers with their decisions, usually in a centralised way. Figure 2.5 shows examples of different management functions.

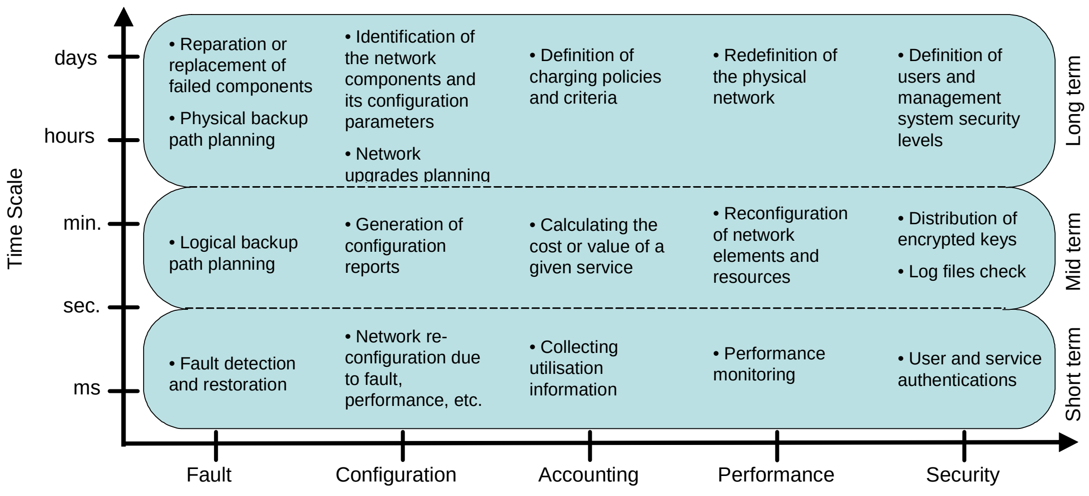

Time Scale
days
hours
min.
sec.
ms
• Reparation or replacement of failed components
• Physical backup path planning
• Identification of the network components and its configuration parameters
• Network upgrades planning
• Definition of charging policies and criteria
• Redefinition of the physical network
• Definition of users and management system security levels
Long term
• Logical backup path planning
• Generation of configuration reports
• Calculating the cost or value of a given service
• Reconfiguration of network elements and resources
• Distribution of encrypted keys
• Log files check
Mid term
• Fault detection and restoration
• Network re-configuration due to fault, performance, etc.
• Collecting utilisation information
• Performance monitoring
• User and service authentications
Short term
Fault
Configuration
Accounting
Performance
Security

Fig. 2.5: Examples of Different Management Functions and Their Time Scale.

Supposing the following assumptions:

- management functions can be considered independent from the physical topology of the system that implements them;
- most of the management functions are independent of each other (e.g. checking log files for security purposes is a completely independent task from fault detection and restoration);
- it is possible to select different architectures for different management functions and combine them into an overall management system (e.g. one function could be implemented in a centralised way while another could be implemented in a distributed way);

then in such case, the interesting point is to find out what the benefits and drawbacks are of the several different ways of implementing the network management functions. This can be called management function architectures. Before we proceed with our classification, we define the following components presented in Table 2.1. These components are used for the definition of network management function architectures.

Symbol	*Name*	*Description*
NE	Network Element	Network element to be managed
A	Simple Agent	Network management agent that monitors a network element and keeps a bunch of variables in a MIB (not confuse with a Software Afent)
HA	Heavy Agent	Complex network management agent capable of receive external code (scripts, mobile agents) and execute network management functions
IM	Intermediate Manager	Simple manager that polls several network management agents and performs simple management functions like filtering, event correlation, etc. on behalf of a manager.
M	Manager	Decision-making application that analyses the collected information, and performs network management functions.
MIB	Management Information Base	MIBs used by network management agents to gather non-elaborated information
MDB	Management Data Base	Data base used by Managers to gather processed information, statistics, etc.

Table 2.1: Network Management Function Architectures Components.

An important point is where the decision-making is placed for a given management function. Usually there are three possibilities: centralised on a single point, equally distributed and partially centralised partially distributed. Another important point is the amount of management information to be sent between hosts. Finally, we also need to consider the consumption of host resources in terms of memory, processing power and data storage. Eventually, there may be other issues to take into account on every specific situation.

The following management function architectures have been identified (Table 2.2): Centralised, Hierarchical and Distributed without Management Centre / Local. Other identified architectures are combinations of the previous three and we grouped them under the term Hybrid architectures. In this group, we specifically describe the following: Distributed with Management Centre, Hierarchically Distributed and Distributed-Centralised.

Network Management Function Architecture	
Centralised	
Hierarchical	
Distributed without Management Centre / Local	
Hybrid:	Distributed with Management Centre
	Hierarchically Distributed
	Distributed- Centralised

Table 2.2: Network Management Function Architectures

A network management function architecture is **Centralised** (Figure 2.6) when the decision-making is concentrated in a single point (M). This central point has a global vision of the whole network and is responsible for 100% of the operating functions: analysis of the collected data from the Agents, filtering, alarm correlation, statistics and derived parameters calculation, etc. When the decision is made, the reconfiguration data must then be sent to all the affected management Agents. The main advantages of this architecture are that it makes it easier to run optimisation operations for the whole network, it enables the use automatic mechanisms but also manual operations performed by human network managers, it is easier to maintain the information integrity and it is secure. The main drawbacks are low scalability in terms of resources consumption (although more resources can be always added and a cluster of manager stations can even be used), delays produced by the polling of many Agents and processing huge amounts of rough information and finally, the lack of robustness (a failure in the management centre could provoke the failure of the whole network).

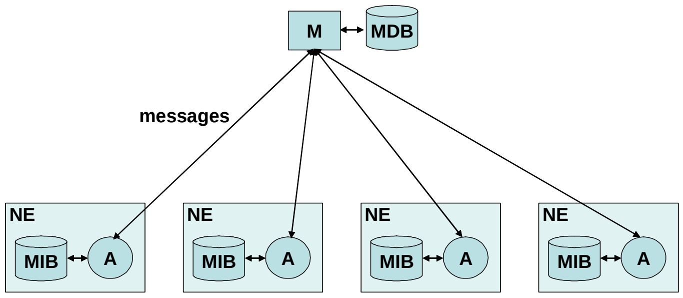


Fig. 2.6: Centralised Function Architecture.

The most suitable management functions for centralised implementation are those that need a global view of the whole network but have an effect in the mid- to long-term, typically analysis, planning and forecasting functions, but also functions that set the operational policies and strategies of the network and configure the network elements and other functions accordingly.

Examples of systems with centralised functions and services are most of the SNMP based management systems and the web service.

The **Hierarchical** function architectures (Figure 2.7) are also centralised in the sense that the decision-making is still concentrated in a single point. The Hierarchical architecture was introduced in order to alleviate the central manager in the task of polling the management Agents and processing the rough data. These tasks are here performed by Intermediate Managers, which are programmed and configured by the Central Manager for polling data from the management Agents. The Intermediate Managers can perform simple operations like summarise the rough data, calculate several parameters, filtering, provide statistics, etc. and the Central Manager receives reports, alarms and events from the Intermediate Managers, i.e. more elaborated information. Usually, all the decisions are made at the Central Manager level and usually it is the Central Manager who directly acts over the Agents.

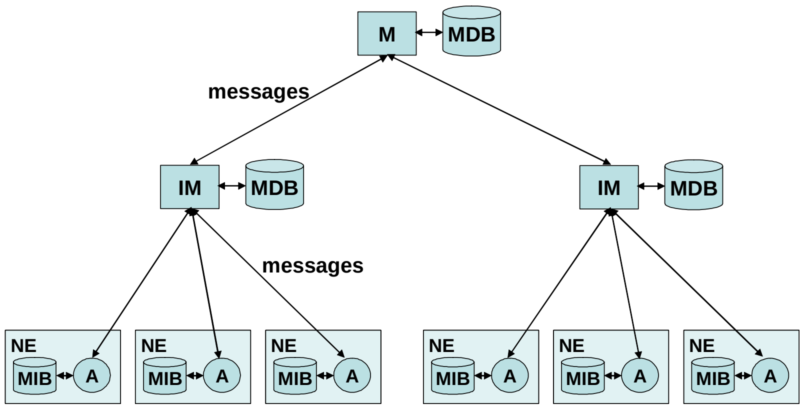


Fig. 2.7: Hierarchical Function Architecture.

This function architecture has the same main advantages as the centralised one, with, in addition, the fact that this system is much more scalable in terms of the load on the Central Manager. It also has the same drawbacks (except the lack of scalability), but adds another: delays increase because it is not the Central Manager who directly picks the information but the Intermediate Managers which then send it up to the Central Manager after processing it. Hierarchical architecture is suitable for the same type of functions as the Centralised architecture, but it is able to manage bigger networks due to its scalability. TMN and SNMP v2/v3 [Siegl and Trausmuth, 1996] compliant management systems follow this Hierarchical architecture and their functions are designed in this way.

At the opposite extreme of the previous two function architectures, there are the **Distributed without Management Centre** (or just Distributed or also called Peer-to-Peer) and **Local** function architectures. These architectures have many similarities and we will explain them together. Both have the main characteristic of having no Central Manager, which implies that the decision-making is equally distributed among the Managers on every Network Element. If these managers collaborate among themselves, exchange management information and base their decisions on this collaboration, then we can talk of a Distributed architecture (Figure 2.8 – A). If these managers act alone, do not communicate with each other for management purposes and their decisions are only based on the local information available on the Network Elements, then this means it is a Local architecture (Figure 2.8 – B).

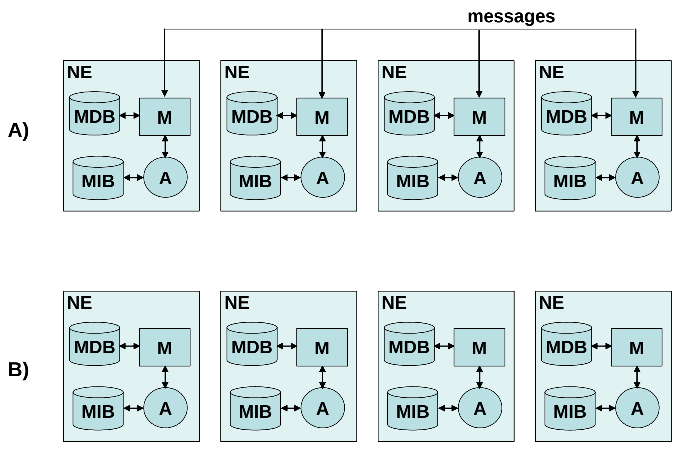


Fig. 2.8: Distributed without Management Centre (A) and Local (B) Function Architectures.

The main advantages arise due to their proximity to the network elements. Figure 2.8 distinguishes between Manager and Agent but they can also be integrated within the same application. Advantages include fast response times, robustness, easy extensibility, etc. The main drawbacks are the lack of a global network view that makes it difficult to use optimisation procedures and the lack of a management centre that makes it impossible for human network managers to collaborate on the management task. Other disadvantages are different in the cases of Distributed or Local architecture. The scalability evaluation in Distributed architectures is difficult. It basically depends on the amount of management information that every Manager needs to send to, or receive from, other Managers in order to be able to make its decisions. With regard to the quantity of this information, the greater it is, the lower the scalability of the architecture. In contrast, Local architecture does not utilise inter-Manager communication and this makes this architecture 100% scalable. However, its lack of communication and global network view makes this architecture very limited and suitable only for very specific cases.

Distributed and Local function architectures are suitable for short term functions, such as fault detection and restoration or performance monitoring. Although there have been several proposals using a Distributed architecture that performs complex functions, many of these proposals are usually based on Distributed Artificial Intelligence (DAI) techniques. Section 2.4. presents a review of these techniques and examples of their use in network management. Most internet routing is performed in a Distributed way. An example of a Local function is presented in [Larsson and Arvidsson, 1999].

Hybrid function architectures combine those of the Centralised, Hierarchical and Distributed. There are many possible combinations and we will just briefly comment on the most significant ones. The first architecture, which we describe as **Distributed with Management Centre** (Figure 2.9), is a development of the Centralised architecture. We use this denomination to encompass two trends: Management by Delegation (MbD) and management based on the use of Mobile Agents. Both have similar characteristics, such as making use of the code mobility property, which can be of two types. In the MbD approach, the Manager downloads the necessary functions (usually scripts) to every Agent, which execute it. Code mobility is only in the Manager-Agent direction [Yemini et al., 1991] [Goldszmidt and Yemini, 1998]. In the Mobile Agent paradigm [Magedanz and Eckardt, 1996] [Bohoris et al., 2000], mobile agents can roam the network collecting information and executing their functions. Therefore, in MbD, the Central Manager only distributes the management tasks to be executed on the Agents (these tasks can be personalised for every Agent), while the Mobile Agents keep the data and execution status and move from Network Element to Network Element performing their functions until they eventually return to the Central Manager. Both these techniques maintain a Management Centre that sends the scripts or mobile agents to the nodes and keeps a global network view. This Management Centre keeps the high level decision-making and network planning, but it distributes the management functions. There usually is a certain degree of decision-making in the Mobile Agents paradigm.

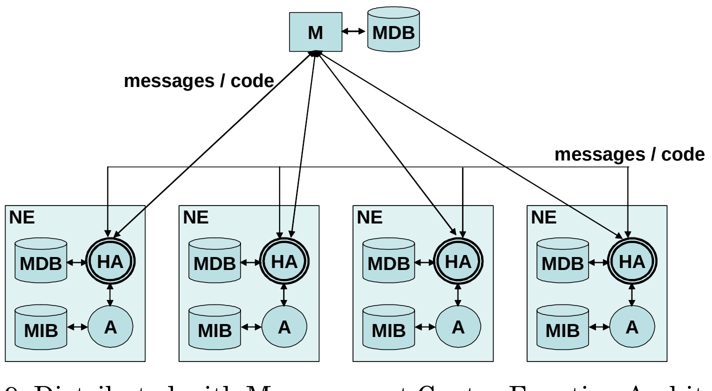


Fig. 2.9: Distributed with Management Centre Function Architecture.

The main advantages of this architecture are its suitability for wireless and mobile networks, where disruptions can be frequent. Also, they make it much easier to introduce new services or modify existing ones. On the other hand, the drawbacks include the fact that this architecture is not useful for high dynamic and re-configurable networks due to the communication overload produced by the need to sent both code and data many times. There are also security problems [Greenberg and Byington, 1998] and the Agent application that receives the MbD scripts or Mobile Agents could be somewhat complex and resource-consuming.

The second Hybrid function architecture we identified is described as **Hierarchically Distributed** and it is a development of the Hierarchical architecture. In this case, the Network Elements are grouped into independent sets, which can be of different sizes. Each set of Network Elements is managed in a distributed way by low level Managers, which take charge of a subset of functions. When a problem arises that spans more than a single set of Network Elements or this problem simply cannot be solved by the low level Managers, then a higher level of Managers takes care of the problem, working also in a distributed way and so on until the top level Manager is reached (Figure 2.10). The top level Manager is responsible for the whole network and maintains a global view.

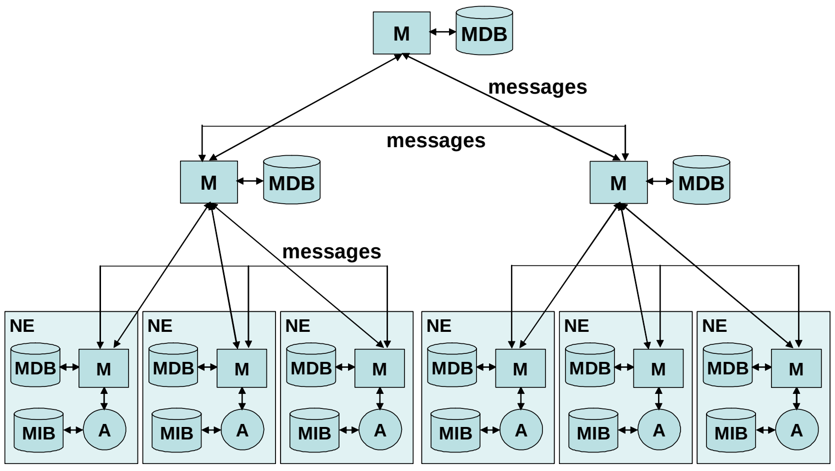


Fig. 2.10: Hierarchically Distributed Function Architecture.

This architecture has the advantages of both the Centralised and the Distributed ones. It was proposed that this kind of architecture was implemented using DAI mechanisms and that the different Managers collaborate and negotiate with each other in order to manage the network. The main advantage of this architecture is its scalability because of the independent sets of Network Elements and the possibility of adding other management layers. The main drawback is its complexity and the difficulty in partitioning the Network Elements. If each set of Network Elements is too small, the actions of management functions always span more than one set, which means upper layer managers end up behaving as in a centralised or hierarchical architecture. If the sets of Network Elements are too big, then the behaviour will be more like a Distributed system. Therefore, it could be difficult to find the balance between these extremes and, once this balance is found, management functions have also to be designed to act in a partitioned way. This architecture was proposed in [Somers, 1996] [Somers et al., 1997]. A similar behaviour (except for the distributed communication between the same level Managers) is followed by the Domain Name Service (DNS), which is highly scalable.

Of course there could also be other possible combinations of Centralised, Distributed and Hierarchical architectures. For instance, a number of Managers can run the network in a Distributed way, without a Central Manager and each one manage a set of Agents in a Centralised way (Figure 2.11), in a **Distributed-Centralised** combination. This is a case similar to the e-mail service where a number of e-mail servers act in a distributed way and each one offers its service to a number of e-mail clients in a centralised way.

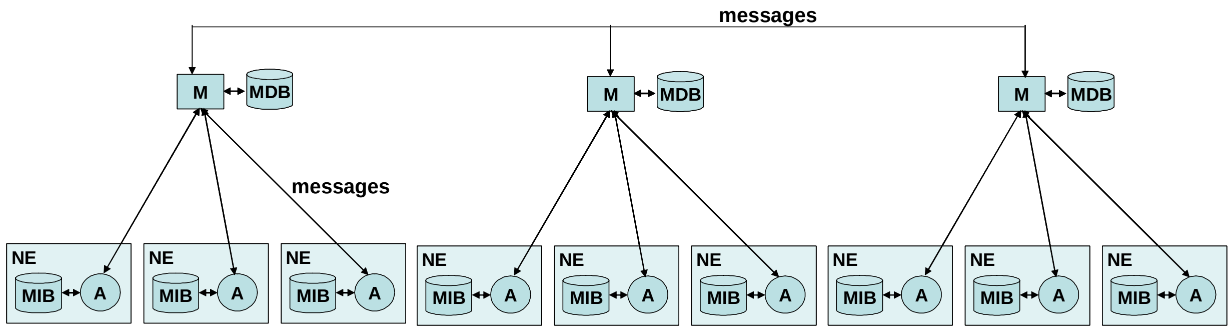


Fig. 2.11: Distributed-Centralised Function Architecture.

The most important conclusion is that it does not matter very much what the system architecture is (it is always, to some extent, distributed) but the function architectures. Different function architectures can be combined on the same management system. Thus, the point is to find out what the best architecture for each management function is. Table 2.3 summarises the characteristics of the management function architectures.

Architecture		*Decision Making*	*Global Network View*	*Enables Human Manager Operation*	*Scalability*	*Robustness*	*Suitable for*
Centralised		Single point	Yes	Yes	Low	Low, need for replication	Mid to long term functions. Optimisation / planning functions. Human interface.
Hierarchical		Single point	Yes	Yes	High	Low, need for replication	
Distributed without Management Centre		Equally distributed with collaboration	No	No	Depends on the degree of collaboration	High	Fast response time functions. Automated functions. Limited to specific cases (Local).
Local		Equally distributed without direct collaboration	No	No	Very high	High	
Hybrid	Distributed with Management Centre (MbD / Mobile Agents)	Single point and multiple points	Yes	Yes, high level	Difficult to evaluate	Medium, need for replication	Low bandwidth, limited connectivity networks. Service management. Updating / upgrading of management function.
	Hierarchically Distributed	Hierarchically distributed	Yes	Yes, high level	High	Medium, need for replication	Automated functions. Functions that can be easily divided into sub-networks / network sections.
	Distributed-Centralised	Equally distributed in a number of nodes	Yes but difficult	Yes but difficult	Depends on the degree of collaboration	High	Fast response time functions. Automated functions. Possibility for optimisation / planning functions.

Table 2.3: Network Management Architectures Summary

2.2.3 Trends in Network Management

Nowadays, network technologies evolve very fast and there is a need for flexibility and dynamism that conventional network management approaches do not have. Network management standards have followed a static/non-extensible client/server model. These systems are huge and hard to design, develop and maintain. Networks have become very complex systems, for the following reasons:

- increasing number of users: more and more users every day, and they are more demanding.
- increasing number of services, many of which (multimedia, interactive, etc.) consume more and more resources.
- increasing network speeds, which lead to greater losses when a network fault occurs.

- increasing competition, network providers want to get the most out of their network resources in order to offer more competitive prices.

The very first consequence of this complexity is that network management systems are also becoming very complex. Due to the large quantity of data to analyse and the dynamism of the networks, network management functions tend to be automated. Therefore human network managers only carry out adjustments, configure and provide directives and restrictions to these automated mechanisms.

Another consequence of this scenario is that not only are the systems becoming more complex, but so are the management functions. This is not only due to the increasing amount of data to process, but also because of the dynamism of their behaviour. For example, Quality of Service (QoS) management has become an important issue; resource management at various levels has also become crucial for performance purposes and also in relation to QoS management, network protection (the avoiding of service disruption when a fault occurs) has also been the subject of a great deal of research.

Automation, complexity and fast response, along with the increasing computing power of the Network Elements, have led to many proposals for using Distributed function architectures [Kahani and Beadle, 1997]. A comparison can be found in [Zhu et al., 2001]. Several of these proposals have been based on DAI techniques and the use of Software Agents. MbD and mainly Mobile Agents [Bieszczad et al., 1998] seems to be ideal for Service introduction and management (customisation and configuration). The ultimate network seems to be self-provisioned with service requests, self-regulated with respect to traffic loads and self-repaired with respect to failures.

2.3 Network Technologies

In this section provide a brief introduction to the main network technologies involved in this work. They are the Asynchronous Transfer Mode (ATM), Multiprotocol Label Switching (MPLS) and Generalised MPLS (GMPLS). This introduction focuses on the key concepts and terminology in order to understand the remainder of this thesis. It is not the intention to give a complete overview of these technologies. Basically, our interest is focused on the ability of these technologies to establish a virtual or logical network that can be used to perform resource management.

2.3.1 Asynchronous Transfer Mode (ATM)

ATM technology has not been as successful as expected, but it is a good technology and much research has gone into it. In many aspects it is still a model to follow. ATM is a network transfer technique capable of supporting a wide variety of multimedia applications with diverse service and performance requirements. It also supports a wide range of traffic bandwidths and traffic types from continuous, fixed-rate traffic to highly intermittent or bursty traffic. Because it can support such a wide range of traffic, ATM was designated as the underlying transport and switching technology for Broadband Integrated Service Digital Network (B-ISDN) [Kim et al., 1994].

ATM is a form of packet-switching technology. That is, ATM networks transmit their information in small, fixed-length packets called "cells", each of which contains 48 bytes of data and 5 bytes of header information (Figure 2.14). It is also connection-oriented, therefore a virtual connection must be established before the transfer of information between two end-points can take place. Finally, as its name indicates, it is asynchronous, which means that time is slotted into cell-sized intervals and slots are assigned to connections in an asynchronous, demand-based manner. This also means that bandwidth is consumed only when some information is transmitted. ATM also gains bandwidth efficiency by being able to statically multiplex bursty traffic sources. There is extensive literature on ATM including tutorials [Le Boudec, 1992] [Sykas et al., 1991] and many books [Kyas, 1995] [Black, 1995].

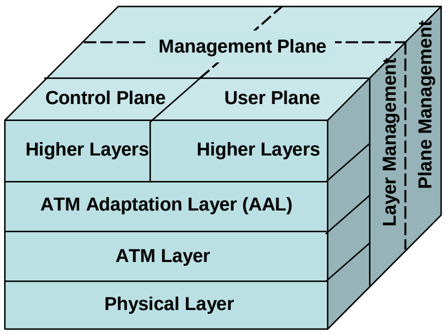


Fig. 2.12: ATM Protocol Reference Model.

The ATM protocol reference model [CCITT Rec. I.321, 1991] is shown in Figure 2.12. The purpose of the protocol reference model is to clarify the functions that ATM networks perform by grouping them into a set of interrelated, function-specific layers and planes. The reference model consists of user, control and management planes. Within the user and control planes is a hierarchical set of layers. The user plane defines a set of functions for the transfer of user information between communication end-points; the control plane defines control functions, such as connection establishment, maintenance and release; while the management plane defines the operations necessary to control the information flow between planes and layers and to maintain accurate and fault-tolerant network operation. Within the user and control planes, there are three layers: the physical layer performing bit level functions, the ATM layer primarily responsible for switching cells and the ATM Adaptation Layer (AAL) responsible for the conversion of higher layer Protocol Data Units (PDU) into ATM cells. Table 2.4 summarises the functions of each layer.

Layer / Sub-layer		*Functions*
AAL	Convergence Sub-layer	Convergence
	Segmentation and Reassembly Sub-layer	Segmentation and reassembly of PDUs of higher protocols.
ATM		Generic flow control Cell header generation/extraction Cell VPI/VCI translation Cell multiplex and demultiplex
Physical Layer	Transmission Convergence Sub-layer	Cell rate decoupling Header Error Control (HEC) Cell delineation Transmission frame adaptation Transmission frame generation/recovery
	Physical Medium Sub-layer	Bit timing Physical medium dependent functions

Table 2.4: Functions of each layer of the ATM protocol reference model.

Across ATM networks, two kinds of interfaces are defined: (i) the User-Network Interface (UNI), which connects an end system to an ATM node and (ii) the Network-Network Interface (NNI), which connects two ATM nodes. Figure 2.13 represents this idea. The cell header differs slightly at these interfaces (Figure 2.14). At the UNI, the header contains a 4-bit Generic Flow Control (GFC) field, used to assist the customer in controlling the traffic flow; a 24-bit label field composed of the 16-bit Virtual Channel Identifier (VCI) and the 8-bit Virtual Path Identifier (VPI), used to identify the Virtual Channel (VC) and Virtual Path (VP) respectively; a 3-bit Payload Type (PT), used to identify whether the payload contains user data or control data; a 1-bit Cell Loss Priority (CLP) field, is used to mark cells as high priority or low priority (low priority cells are dropped before high priority cells when congestion occurs); and a 8-bit Header Error Check (HEC), which is a CRC polynomial that protects the cell header. The NNI cell header is identical except that it does not have the GFC field. These four bits are added to the VPI field so, in this case, the VPI has 12 bits. In addition, the AAL layer, responsible for mapping the requirements of higher layer protocols onto the ATM network, operates only in ATM devices at the edge of the ATM network and is totally absent in ATM switches since the switching occurs at the ATM layer. This is shown in Figure 2.15.

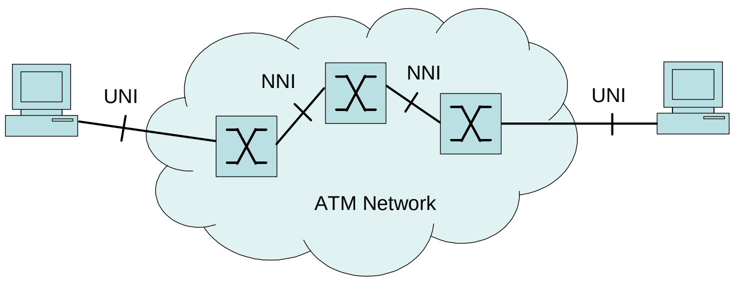


Fig. 2.13: Example of an ATM Network with UNI and NNI Interfaces.

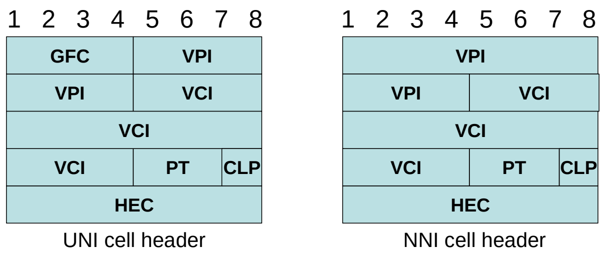


Fig. 2.14: UNI and NNI 5-byte-long Cell Headers.

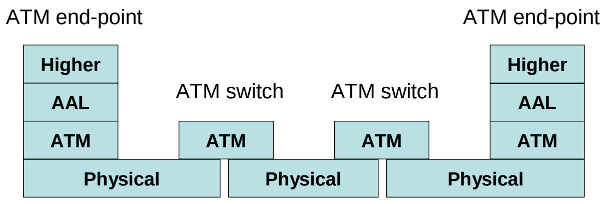


Fig. 2.15: End-systems Layers and Switches Layers.

A VC connects two ATM end-points, while a VP connects any two ATM devices, including both switches and end-points. Several VCs can be multiplexed onto the same VP. When cells arrive at ATM switches, their VPI/VCI values are examined in order to determine the output port and to forward the cell. In this process, the VPI/VCI input values are translated to new VPI/VCI output values using a table that was initialised during the connection establishment. Thus, an end-to-end VC will usually have several VPI/VCI values during its journey. If an ATM switch translates only VPI values, it is called a “VP switch” and if it can translate both VPI and VCI values it is called “VP/VC switch”. Therefore VPI/VCI values do not represent a unique end-to-end virtual connection and they can be reused at different switches throughout the network (Figure 2.16).

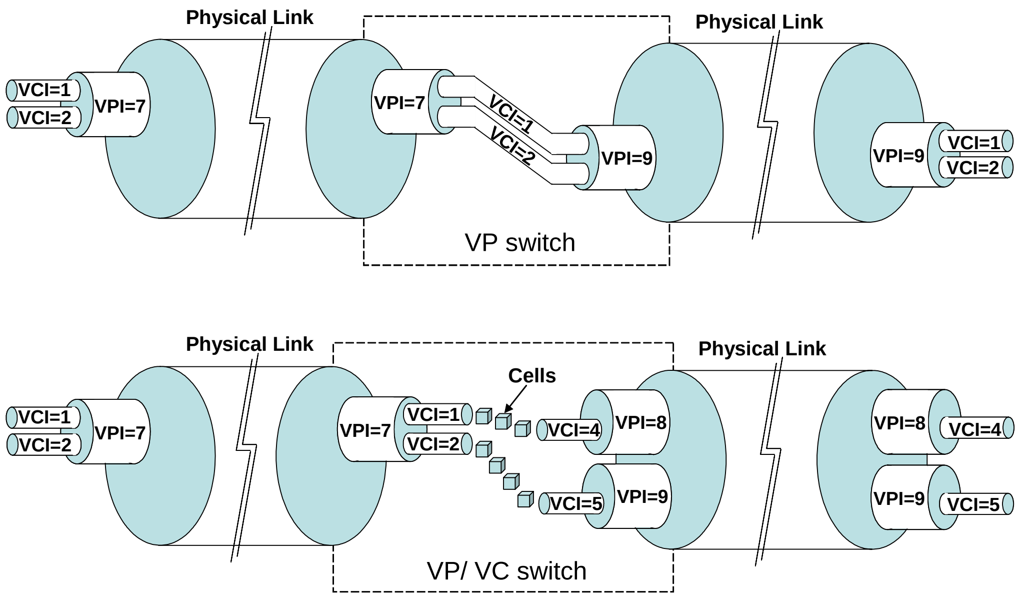


Fig. 2.16: VP Switching and VP/VC Switching.

In consequence, the connections in an ATM network are organised into a two-level hierarchy of channels and paths. Cells transmitted over a single physical link are associated to a particular connection by VPI/VCI fields in the cell header. A channel (the smallest unit) is identified using both VPI and VCI values, while a path (a group of channels) is identified by its VPI only.

The structure of connections in a two level hierarchy has a number of implications and advantages. In a VP switch, the size of the forwarding table is kept relatively small, even for a large number of connections. Another consequence is the possibility of accommodating Virtual Private Networks (VPN). VPs and VCs can be established with any amount of allocated capacity, even zero. Therefore VPs can also be used as pre-reserved, end-to-end paths. The main advantage of using them in this way, is that it simplifies the establishment of VCs (since the routing and reservation has already been carried out).

As ATM has been designed to transport a wide range of different services (including interactive, multimedia, distribution and traditional data services, etc) and all these services have different QoS requirements, the user establishes a traffic contract with the ATM network that guarantees the parameters of the contract. The traffic contract is based on three main performance measures describing the QoS of a particular connection: Cell Loss Rate (CLR), Cell Transfer Delay (CTD) and Cell Delay Variation (CDV).

There are many mechanisms in ATM for guaranteeing traffic contracts with users and the required QoS for the different services. First, there is the connection-orientated form of transmission over a virtual circuit that preserves the cell sequence integrity. Secondly, the services are grouped into a small number of service categories that have homogeneous characteristics in terms of traffic patterns, QoS requirements and possible use of control mechanisms. Both ATM forum and ITU-T have identified these service categories [ITU-T Rec. I.371, 1995] [ATM Forum, 1996]. Broadly speaking there are guaranteed services (Constant Bit Rate or CBR, Variable Bit Rate or VBR) with fixed traffic descriptors and services with a minimum guarantee of traffic requirements (Available Bit Rate or ABR) that adapt dynamically, using congestion control mechanisms. There is also an non-guaranteed service (unspecified bit rate or UBR) where no traffic descriptors are used and no QoS guarantees are made. In the third case, there are several control functions that act directly on the cells in the network in order to influence and protect the QoS of the accepted connections, including preventive and reactive control mechanisms. There are other more specific service categories.

The main preventive mechanisms include Connection Admission Control (CAC) and Usage Parameter Control (UPC). CAC is responsible for determining whether a connection request should be accepted or rejected by the network during the connection set-up phase [Marzo, 1996]. One of the most common ways for an ATM network to make a connection admission decision is to use the connections traffic descriptors and QoS requirements to predict the "equivalent bandwidth" [Guerin et al., 1991] required by the connection. The equivalent bandwidth determines how many resources need to be reserved by the network to support the new connection at its requested QoS. UPC deals with the already accepted connections, ensuring that they do not exceed the traffic parameters declared at connection set-up. UPC usually uses a leaky bucket mechanism and it can take various actions if a source is violating its contract. It can delay violating cells or simply drop them, it can change their CLP bit (in the header of the cell) so that they can be dropped if required by the intermediate nodes along the virtual channel, or UPC can inform the source when it starts to violate its contract.

Reactive control functions are useful for service classes such as ABR, which allows sources to use bandwidth not being utilised by calls in other service classes. This is done using special control cells called resource management cells. The first possibility is a source-destination feedback loop where the intermediate switches mark their current congestion state. When the cells return to the source, it adjusts the rate according to whether the network is congested or under-utilised. The second possibility is a link-by-link traffic control where intermediate switches exchange resource management cells with the buffer space available to the next switch downstream.

ATM follows the principle of out-of-band signalling, therefore signalling and data channels are separated. Its main purpose is to establish, maintain and release ATM virtual connections. Typically certain VPI and VCI values are reserved by ATM networks for signalling messages.

ATM has also defined Operations and Maintenance (OAM) functions in order to ensure high-quality and reliable VPs and provides fault management and performance monitoring mechanisms using special cells called OAM cells [Yahia and Robach, 1997] [Chen and Liu, 1994]. These cells have the same VPI field value as user cells, but are distinguished by special VCI values and PT field values. Figure 2.17 shows the structure of an OAM cell.

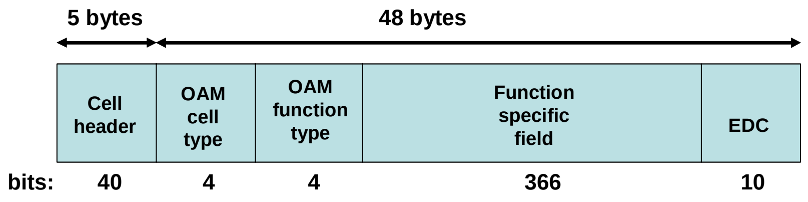


Fig. 2.17: OAM Cell Structure.

For instance in [Sato et al., 1991], the OAM cells are used for performance monitoring. They are initiated during or after connection set-up. OAM cells for both directions must follow the same physical route, so that any connecting points can monitor the OAM information from both directions. Both endpoints and connecting points can generate, insert, monitor and process OAM cells for a given VP; only endpoints can extract OAM cells. Fault management is performed by three methods: surveillance of alarms from the physical layer, continuity checking and on-demand testing of connectivity by cell loopback. OAM cells are proposed in [Chen et al., 1996] as the key mechanism for a distributed control of VPs and VP networks.

2.3.2 Multiprotocol Label Switching (MPLS)

Multiprotocol Label Switching [Rosen et al., 2001] [Davie and Rekhter, 2000] can be seen as a more flexible approach to the deployment of connection-orientated networks. Most network resource management research in ATM can be translated to MPLS without major changes, e.g. [Leon-Garcia and Mason, 2003]. It is an IETF initiative [IETF URL] and nowadays is widely deployed in networks worldwide.

MPLS is an advanced forwarding scheme that enables management mechanisms for the core network belonging to a network provider, usually in an Internet environment. MPLS groups user flows into aggregates and allows a certain capacity to be allocated to each aggregate. Its main characteristic is the separation of an IP router's functions into two parts [Chen and Oh, 1999]: the forwarding plane, responsible for how data packets are relayed between IP routers using a label swapping mechanism; and the control plane, consisting of layer routing protocols to distribute routing information between routers, and label binding procedures for converting this routing information into the forwarding tables needed for label switching [Rosen et al., 2001]. This is shown in Figure 2.18.

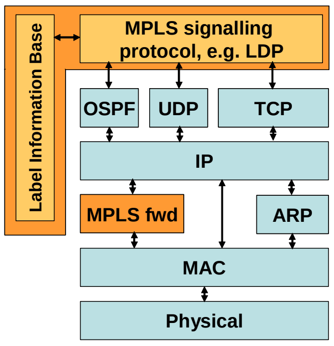


Fig. 2.18: Control and Forwarding Components.

Routers belonging to an MPLS domain are called Label Switched Routers (LSR). When a data packet comes into an MPLS domain, through an ingress LSR, the packet is classified into a specific Forwarding Equivalent Class (FEC). A FEC groups the packets with certain common properties (protocol, size, origin, destination, etc.). These packets are then routed according to a combination of this information carried in the IP header of the packets and the local routing information maintained by the LSR. An MPLS header is then inserted for each packet.

MPLS is intended to run over multiple data link layers, such as:

- ATM, where the Label is contained in the VPI/VCI field of the ATM header.
- Frame Relay, where the Label is contained in the DLCI field in the FR header.
- PPP/LAN, where the MPLS header is inserted between the layer-two and layer-three headers.

In a non-ATM or FR environment, the MPLS header contains a 20-bit Label, a 3-bit experimental (Exp) field, formally called Class of Service (CoS), a 1-bit Label Stack Indicator (LSI) and a 8-bit Time To Live (TTL) field. Figure 2.19 shows the MPLS header. An LSR examines the Label and possibly the Exp field for forwarding the packet. Each LSR use the Label as the index to look up the forwarding table. The incoming Label is replaced by the outgoing Label and the packet is switched to the next LSR. This Label-switching process is very similar to the VPI/VCI processing in the ATM switches. Before a packet leaves the MPLS domain, its MPLS header is removed.

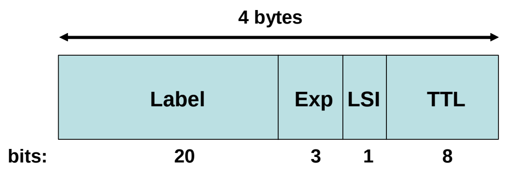


Fig. 2.19: MPLS Header.

The packets inside an MPLS domain go from an ingress LSR to an egress LSR along Label Switched Paths (LSP). MPLS has been designed to work with existing Internet routing protocols such as Open Shortest Path First (OSPF) [Moy, 1998], Border Gateway Protocol 4 (BGP-4) [Rekhter and Li, 1995], or Intermediate System to Intermediate System (IS-IS) [Oran, 1990]. Explicit routing can also be utilised. This separation of the fast forwarding and the routing mechanisms enables each component to be developed and modified independently. In order to set up the LSPs, MPLS utilises a signalling protocol such as Resource Reservation Protocol (RSVP) [Braden et al., 1997] and Label Distribution Protocol (LDP) [Anderson et al., 2001], or the RSVP-TE and Constraint-based Routing LDP (CR-LDP) [Jamoussi et al., 2002], which are extensions to support some Traffic Engineering (TE) capabilities. An example of an MPLS domain is depicted in Figure 2.20.

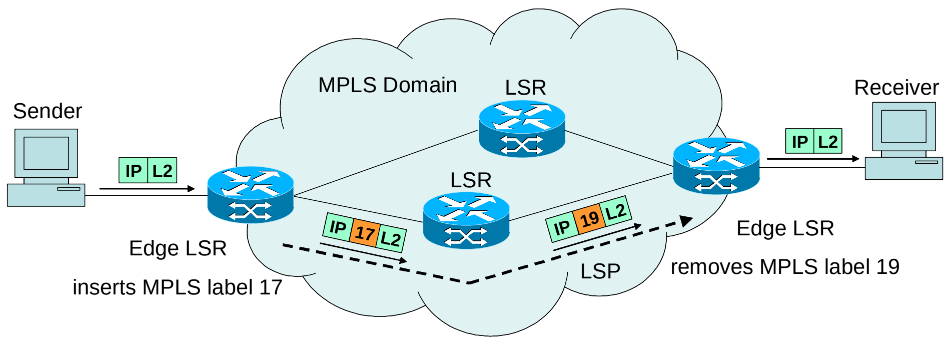


Fig. 2.20: MPLS Domain.

An interesting property of MPLS is that the LSI bit allows stacking MPLS Labels [Rosen et al., 2001b]. A Label stack is an ordered set of labels appended to a packet. This enables MPLS tunnelling. The LSI bit is set to 1 for the last entry in the label stack (i.e. for the bottom of the stack) and 0 for all other label stack entries. Figure 2.21 shows this procedure. Note that only the top label of the label stack is processed in the LSRs.

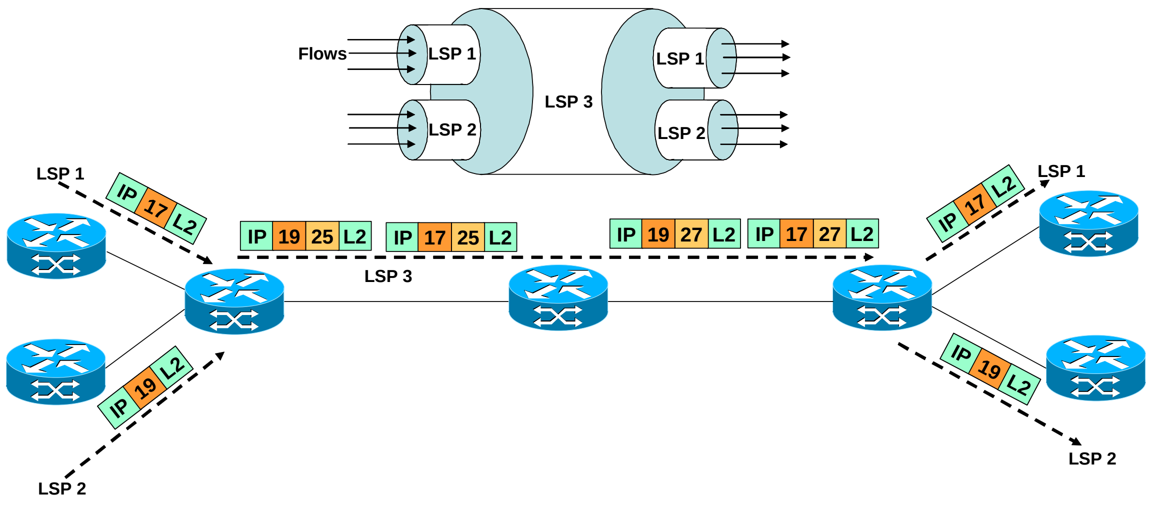


Fig. 2.21: LSP Tunnelling.

In order to control the path of LSPs effectively, each LSP can be assigned one or more attributes. These attributes will be considered for computing the path [Awduche et al., 1999]. Table 2.5 summarises these attributes.

Attribute Name	*Description*
Bandwidth	The minimum requirement for reservable bandwidth of a path for the LSP to be set up along that path.
Path	It decides whether the path of the LSP should be manually specified or dynamically computed by constraint-based routing.
Setup Priority	It decides which LSP will get the resource when multiple LSPs compete for it.
Holding Priority	It decides whether an established LSP should be pre-empted of the resource it holds by a new LSP.
Affinity	It can be used to set the LSP affinity (positive or negative) to certain network links.
Adaptability	Whether to switch the LSP to a more optimal path when one becomes available.
Resilience	It decides whether to re-route the LSP when the current path is affected by failure.

Table 2.5: LSP Attributes.

MPLS itself cannot be used in order to guarantee QoS in the Internet. Although it can be used to establish paths and make reservations, it cannot be used for admission control for instance. MPLS has to be combined with other schemes, such as Differentiated Services (Diffserv) [Blake et al., 1998]. Users and Internet Service Providers (ISP) establish Service Level Agreements (SLA) between each other. At the ingress of the ISP, networks packets are classified, policed and possibly shaped accordingly. Using these classifications, policing, shaping and scheduling mechanisms, ISPs can provide different services, e.g. premium, assured and best-effort services. SLAs can be static (negotiated monthly or yearly) or dynamic. Dynamic SLAs allow customers to request services on demand, without subscribing to it. Signalling and admission control are needed in this case.

MPLS, along with Constraint-based routing, TE mechanisms and the use of Diffserv, is a suitable framework for providing QoS guaranteed services in the Internet [Xiao, 2000]. Constraint-based routing computes routes that are subject to constraints, such as bandwidth and administrative policy. Because constraint-based routing considers more than the minimum path (in terms of hops) in computing routes, it may find a longer but more lightly loaded path better than a heavily loaded shortest path. Constraint-based routing can be online (routers may compute paths for LSPs at any time) or offline (an offline server computes paths periodically) [Marzo et al. 2003b]. In order to make constraint-based routing possible, an enhanced link state Interior Gateway Protocol (IGP) is required. This enhanced IGP must be used to propagate link attributes in addition to normal link state information and also flood information more frequently than a normal IGP. Signalling protocols, RSVP-TE and CR-LDP and routing protocols, OSPF-TE and IS-IS-TE, support TE [Ashwood-Smith and Berger, 2003] [Berger et al. 2003a] [Berger et al. 2003b].

An additional feature of an MPLS-based service architecture are the class-based routing with LSPs. Sometimes, it is desirable to let different classes of traffic take different paths. In order to do so, multiple LSPs can be created from the same source to the same destination, thus LSPs for different classes can have different constraints [Xiao, 2000]. This is done in a similar way than in ATM networks, using different VPs for different classes [Dziong, 1997].

2.3.3 Generalised MPLS (GMPLS)

Current and future data and transmission networks will consist of elements such as packet routers, ATM switches, Dense Wavelength Division Multiplexing (DWDM) systems, Add-Drop Multiplexers (ADMs), Photonic Cross-Connects (PXCs), Optical Cross-Connects (OXCs), etc. GMPLS [Mannie, 2003] [Banerjee et al., 2001], an IETF initiative [IETF URL], provides a common control plane for packet, TDM (Time Division Multiplexing) and wavelength services. It can be used to dynamically provision resources and to provide network survivability in many different network technologies.

GMPLS differs from traditional MPLS in that it supports multiple types of switching. The original MPLS architecture is extended to include LSRs whose forwarding plane recognises neither packet, nor cell boundaries and therefore, cannot forward data based on the information carried in either packet or cell headers. This new set of LSRs, or more precisely interfaces of these LSRs, can be divided into the following classes:

- Packet Switch Capable (PSC). Interfaces that recognise packet boundaries and can forward data based on the packet header (IP header, MPLS header, etc).
- Layer-2 Switch Capable (L2SC). Interfaces that recognise frame/cell boundaries and can switch data based on the content of the frame/cell header (MAC header on Ethernet, ATM cells header, etc).
- TDM. Interfaces that switch data based on the data's time slot in a repeating cycle (SONET/SDH cross-connects, ADMs, etc).
- Lambda Switch Capable (LSC). Interfaces that switch data based on the wavelength on which the data is received (PXC, OXC, etc).
- Fibre Switch Capable (FSC). Interfaces that switch data based on a position of the data in the physical space (PXC, OXC that operate at the level of single or multiple fibres).

An LSP can be established only between, or through, interfaces of the same type. Depending on the particular technology being used for each interface, different circuit names can be used (e.g. SDH circuit, optical trail, light-path, etc.). In GMPLS they are all referred to as LSPs.

LSP hierarchy is the notion that LSPs can be nested inside other LSPs, giving rise to a hierarchy of LSPs. This is achieved by considering an LSP as a link or virtual link. This is already possible in MPLS. This hierarchy can occur on the same interface (using the Label Stack / tunnelling concepts) and also between different interfaces. For instance MPLS LSPs that enter the optical transport domain at the same node and leave the domain at the same node may be aggregated and tunnelled within a single optical LSP. This aggregation helps to conserve the number of lambdas used by the MPLS domain.

LSP hierarchy also helps deal with the discrete nature of optical bandwidth. When an optical LSP is set up, it gets a discrete bandwidth (say 2.488 Gbps). However, when this optical LSP is treated as a link, that link's bandwidth need no longer be discrete. A 100 Mbps MPLS LSP crossing the optical transport domain can be tunnelled through the optical LSP, leaving 2.388 Gbps for other MPLS LSPs. Allocating an entire 2.488 Gbps for every MPLS LSP that crosses the optical domain would be impractical. A natural hierarchy exists that dictates the order in which LSPs can be nested. This hierarchy is based on the multiplexing capability of the LSP types. At the top of this hierarchy, there are nodes that have FSC interfaces, followed by nodes that have LSC interfaces, followed by nodes that have TDM-capable interfaces, followed by nodes with L2SC interfaces, followed by nodes with PSC interfaces (Figure 2.22).

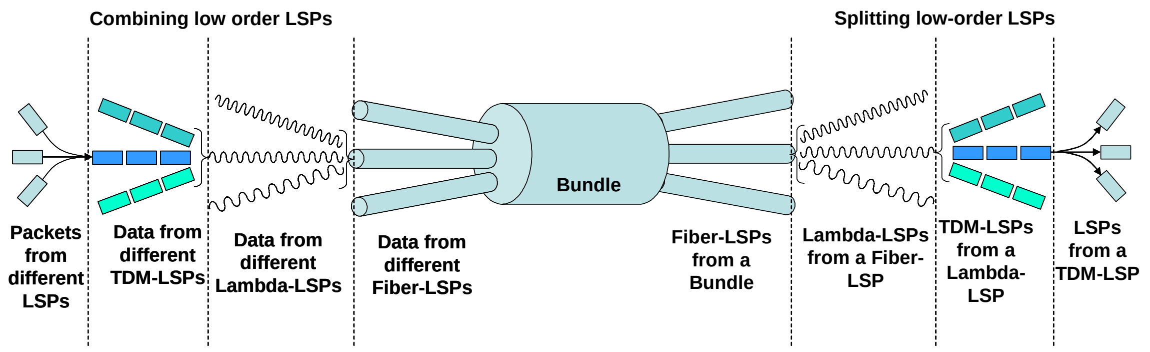

Combining low order LSPs
Splitting low-order LSPs
Bundle
Packets from different LSPs
Data from different TDM-LSPs
Data from different Lambda-LSPs
Data from different Fiber-LSPs
Fiber-LSPs from a Bundle
Lambda-LSPs from a Fiber-LSP
TDM-LSPs from a Lambda-LSP
LSPs from a TDM-LSP

Fig. 2.22: LSP Hierarchy in GMPLS.

GMPLS architecture clearly separates the control plane and the forwarding plane (non packet-based forwarding planes are now considered). In addition, it also clearly separates the control plane into two parts: the signalling plane, containing the signalling protocols, and the routing plane, containing the routing protocols. These routing and signalling protocols are based on well-known signalling and routing protocols that has been extended and/or modified to support GMPLS: two signalling protocols, RSVP-TE and CR-LDP and two routing protocols, OSPF-TE and IS-IS-TE, have been extended [Ashwood-Smith and Berger, 2003] [Berger et al., 2003a] [Berger et al. 2003a]. They use IPv4 and/or IPv6 addresses. Only one new specialised protocol is required to support the operations of GMPLS, a signalling protocol for link management called Link Management Protocol (LMP) [Lang, 2003]. LMP was designed to manage the data links, independently of the termination capabilities of those data links.

As a consequence of the separation of the planes, it is possible to use one physical network for data transmission and another, different one, for control. This idea is shown in Figure 2.23. In GMPLS, the control channels between two adjacent nodes are no longer required to use the same physical medium as the data-bearing links between those nodes. For example, a control channel could use a separate wavelength or fibre, an Ethernet link, an IP tunnel through a separate management network, or a multi-hop IP network.

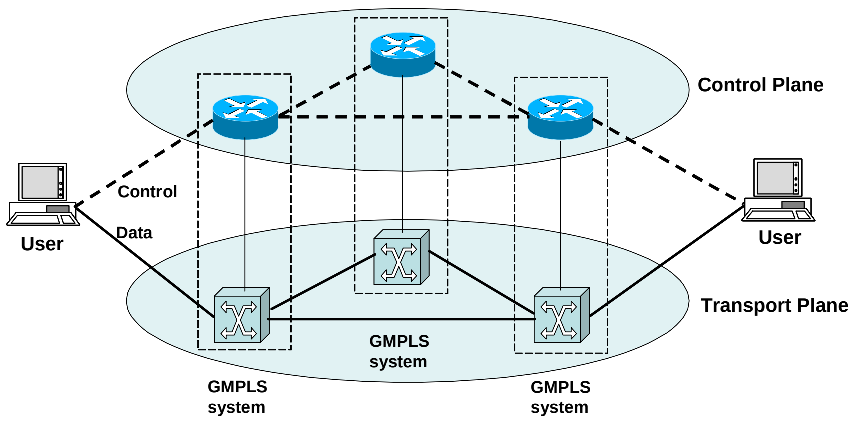


Fig. 2.23: GMPLS Plane Separation.

Link Management Protocol. The new LMP can also be utilised to manage and maintain the control and data planes between two neighbouring nodes. LMP is an IP based protocol that includes extensions to RSVP-TE and CR-LDP. LMP provides four basic functions for a node pair: control channel management, link connectivity verification, link property correlation and fault isolation. Control channel management is used to establish and maintain connectivity between adjacent nodes and consists of a lightweight, keep-alive Hello protocol that is transmitted over the control channel. The link verification procedure is used to verify the physical connectivity of the component links. The Link Summary message of LMP provides the correlation function of link properties (e.g., link IDs, protection mechanisms and priorities) between adjacent nodes. Finally, LMP provides a mechanism to isolate link and channel failures, independently of the data format. Control channel management and link property correlation are mandatory procedures for LMP.

2.4 Software Agents in Telecoms

This section introduces Software Agents, the Distributed Artificial Intelligence (DAI) notion of Agents. Confusion can arise because in both network management and DAI, the same word, "Agent", is used for different purposes. The term "Software Agents" is used in this work for the DAI Agents. After this brief introduction there is a review of several examples of how Software Agents have been applied to network management.

2.4.1 Software Agents

A Software Agent is a computer entity (independently executing program) that is capable of acting autonomously. Software Agents can differ in abilities, but typically they possess a certain degree of expertise to deal with their own world. Probably the most important attribute that distinguishes Software Agents from other types of software processes is their ability to cooperate in some manner with other agents (human or software), either to gain a local advantage themselves or to add to the global "good" of the system in which they operate.

The properties of Software Agents are described in many works, such as [Nwana, 1996], but a key reference is [Wooldridge and Jennings 1995]. This work listed there properties as follows:

- Autonomy: agents operate without the direct intervention of humans or others and have some kind of control over their actions and internal state.
- Social Ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language.
- Reactivity: agents perceive their environment (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined) and respond in a timely fashion to changes that occur in it.
- Pro-activeness: agents do not simply act in response to their environment; they are able to exhibit goal-directed behaviour by taking the initiative.

While this list of properties may be added to, it represents the minimum that a software entity needs to be considered a software agent. Others (i.e. learning) can drive us to a more solid notion of agency. [Luck et al. 2003] describes the current situation of software agent technology and its perspectives for the future.

Software Agents can be classified by different criteria. Based on their internal architecture, we can distinguish between reactive, deliberative and hybrid agents. The **reactive** agents act simply, using a stimulus/response type of behaviour by responding to the current state of the environment in which they are embedded. **Deliberative** agents act in a more reflexive way and usually contain a symbolic model of the world. Deliberative agents are often viewed in terms of beliefs, desires and intentions in the so-called BDI architecture. Finally **hybrid** agents combine the reactive and deliberative behaviours.

A Multi-agent System (MAS) can be defined as a set of agents that interact with each other and with the environment to solve a particular problem in a co-ordinated (behaviourally coherent) manner. MAS is one of the most important sub-fields of Distributed Artificial Intelligence (DAI). The MAS concept is largely based upon the idea that complex activities are the outcome of interactions between software agents. A good reference on MAS is [Weiss, 1999].

Agents in a MAS can also be classified by their behaviour into co-operative agents, self-interested agents or hostile agents. The arguments in favour of a certain class are mostly weak and the advantage of one depends on the problem faced. A particular case is the market-based approach, where a collective behaviour is obtained from the interaction of independent, self-interested agents in an economic market environment. Usually, they utilise economic mechanisms such as auctions, bidding, pricing, etc. and everything of interest to an agent is described in terms of prices. There are two types of agents, consumers and producers and they bid for goods at various prices in order to maximise either their profits or their utility until an equilibrium is reached [Weiss, 1999].

The individual agents on a MAS are usually organised in a what is called MAS architecture. A particular characteristic of a MAS is whether all the agents are of the same or different type. Another key issue concerning Software Agents is their communications. To enable the agents' interaction in a MAS, it is essential to promote and facilitate rich and precise inter-agent communication. In order to achieve this rich communication, an Agent Communication Language (ACL) is needed. The ACL describes the way agents exchange messages. It is very advisable to use standardized ACLs. The two most standardized ACLs are the Knowledge Query and Manipulation Language (KQML) and the very similar Foundation for Intelligent Physical Agents (FIPA) ACL. FIPA is a very active organisation dedicated to producing standards for heterogeneous and interacting agents and agent-based systems [FIPA URL].

A remarkable trend in agent technology involves the Mobile Agents. They extend the concept of code mobility. While the initial idea of executing the agent code where needed and updating the "intelligence" easily, is quite simple and has several obvious benefits, its implementation presents several difficulties. The Mobile Agent paradigm currently has two general goals: reduction of network traffic and asynchronous interaction [White, 1997]. They are capable of travelling through networks, interacting with foreign hosts, gathering information on behalf of their owner and returning home after performing their tasks. Particularly interesting is the ability of mobile agents to deal with the problem of having a non-continuous connection to networked resources. This is especially relevant in non-robust networks (or not protected networks), wireless communications or mobile clients.

2.4.2 Agents as Network Management Systems

Software agents are therefore a key technology when it comes to proposing solutions for complex systems. Moreover, Software Agents are inherently distributed and thus they are also a suitable approach for distributed problems. Hence, it seems that Software Agent technology is suitable for management systems in general, and for network management systems in particular [Magedanz et al., 1996] [Weihmayer and Velthuijsen, 1998] [Muller, 1997].

As network and service management has become a very complex distributed task, it is a natural area in which to apply Software Agents. Their utilisation is open to any network management system (i.e. telephony networks, cellular networks, data communications networks, etc) as an automation element, but also enriched by DAI mechanisms in order to perform decentralised management.

Decentralised management leads to the delegation of tasks. Delegation is a generic word to describe the process of transferring power, authority, accountability and responsibility for a specific task to another entity [Martin-Flatin and Znaty, 2000]. A complementary relational structure is co-operation/competition. A typical network management system based on intelligent agents will be made up of one of or a mix of these organisational structures. Figure 2.24 depicts a hierarchy tree representing the delegation and co-operation/competition relationships.

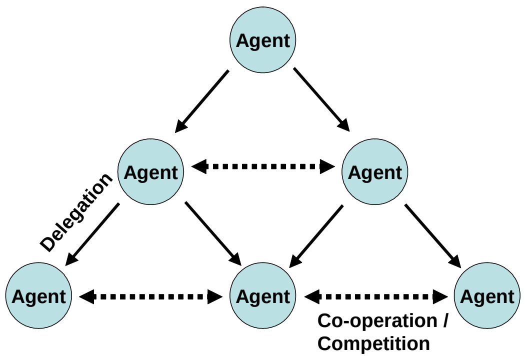


Fig. 2.24: Delegation and Co-operation/Competition.

There are an enormous number of examples of Software Agent proposals in network management, including both static and mobile MAS. It is not possible to review all the work here done, proposal by proposal. More detailed reviews on intelligent agents for telecommunications management can be found in the Intelligent Agents for Telecommunication Applications series [Albayrak, 1998], [Albayrak, 1999] and [Albayrak and Garijo, 1998]; in books [Hayzelden and Bigham, 1999a] and [Hayzelden and Bourne, 2001]; and articles [Cheikhrouhou et al. 1998], [Hayzelden and Bigham, 1999b] and [Vilà, 2001]. On the other hand, it should be noted that each proposal usually tries to address different problems and that publications give only main ideas and general descriptions and that it is extremely difficult to reproduce the systems. As a consequence, the information available on the different systems is highly diverse and comparison becomes very difficult.

Static MAS are suitable for most networks, but usually utilised in reliable high-capacity core networks. Examples of static agent systems applied to network management are given below:

- In the British Telecom works [Davison et al., 1998] and [Hardwicke and Davison, 1998], a MAS is proposed to perform a dynamic bandwidth management in ATM networks. This MAS uses two sets of simple agents: one set is in charge of the bandwidth change and the other set is in charge of the VP re-routing. The two sets of agents are completely independent and there is no direct communication between them.
- In the HYBRID project [Somers, 1996] [Somers et al., 1997], a well-defined architecture based on a geographical hierarchical structure is proposed (Figure 2.25). This approach has three layers: local, regional and national, each layer is responsible for a particular region of the network. Upper layers delegate the management to lower ones and when a problem occurs and the lower layers cannot solve it, a message is sent to the upper layers. This approach uses many different deliberative agents comprising all the management functions (FCAPS) and it proposes the complete substitution of any other management system.

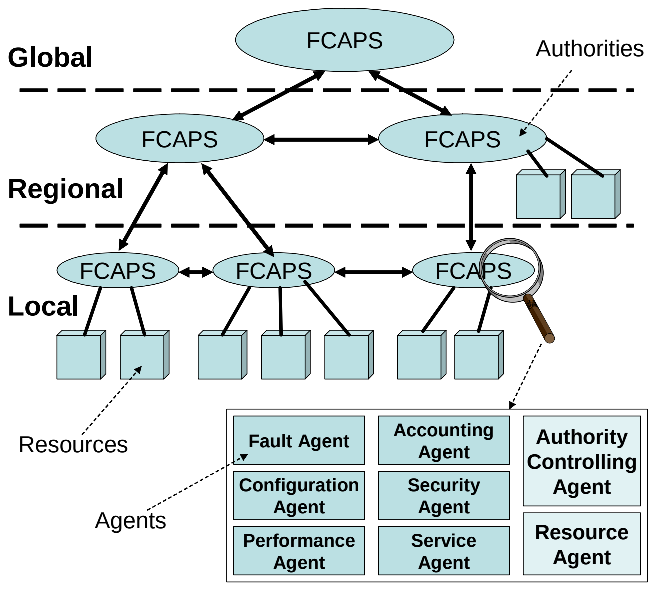


Fig. 2.25: HYBRID Architecture.

- A MAS approach based on multiple layers of agents is proposed in Tele-MACS architecture [Hayzelden, 1998] [Hayzelden and Bigham, 1998], which integrates distributed low-level reactive agents and pro-active planning agents (Figure 2.26). The agent system subsumes the proprietary control system with the purpose of providing more intelligent control. One of the main goals of this system is the CAC.

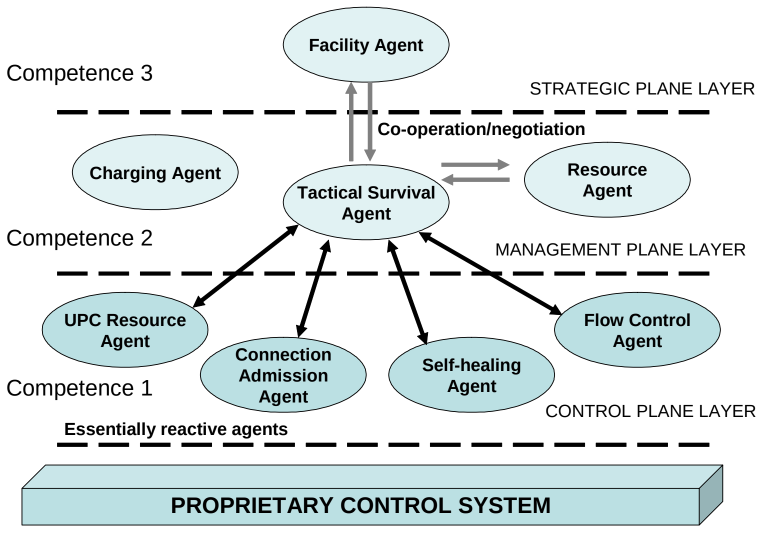


Fig. 2.26: Tele-MACS Architecture.

- The ACTS (Advanced Communication Technologies and Services) project IMPACT [Luo et al., 1999] [Bigham et al., 1999] presents a more complex scenario where a multi-layered MAS is designed to carry out a complete ATM resource management including CAC, routing, dynamic bandwidth management, multiple Service Provider support, etc. The idea is to demonstrate that agent based control can act within the constraints of traditional connection admission procedures in a real ATM test-bed.
- Other works, in connection with Static MAS used for network management are presented in Table 2.6.

Static Software Agents could be a valid solution to many network management issues but the addition of mobility could contribute by providing several benefits. Mobility is a property that can be added to the agents so that they can move between hosts to arrive at the network element which is object of management and interact with it, taking advantage of the local interaction. Mobile Agents can save space and processing capabilities, reduce network traffic, allow asynchronous autonomous interactions, facilitate interaction with real-time systems, add robustness and fault tolerance, give support for heterogeneous environments and facilitate software upgrades and extensibility [Bieszczad et al., 1998][Susilo et al., 1998]. By reducing the length of the control loop, management applications gain much faster response times and reliability [Goldszmidt and Yemini, 1998]. The performance advantage of migrating the agent (acting as a client) and getting it to work locally in the network element (acting as a server), depends partly on the network: the lower its throughput or availability, or the higher its latency or cost, the greater the advantage [Bohoris et al., 2000]. An advantage is that Mobile Agents provide flexibility in extending the functionality offered by existing communication systems and a drawback is the increased complexity of the security infrastructure in order to avoid a mobile agent acting like a virus or a worm [Zapf et al. 1998].

References	*Description*
[Cheikhrouhou, 1999] [DIANA URL] [Oliveira et al., 1996]	"Management Awareness in Networks with Intelligent Agents" (MANIA) and "Distributed Intelligent Agent for Network Administration" (DIANA) projects. Hierarchical management, where the objectives are split and dispatched to the different layers of agent managers, which can delegate sub-goals into basic agents.
[Eurescom P712 URL] [Corley et al., 2000]	Project P712 called "Intelligent and mobile agents and their applicability to service and network management". Cases study including the investigation into how agents could enable customers, service providers and network providers to negotiate automatically for services and network resources.
[Bodanese and Cuthbert, 1999]	Resource management of mobile telephony networks. Use of hybrid agents for a distributed channel allocation scheme.
[Gibney et al., 1999]	Market-based call routing based on self-interested agents representing network resources without assuming a priori co-operation
[Willmott and Faltings, 2000]	On-line QoS call routing in ATM networks. Hierarchical structure with a controller agent responsible for resource allocation decisions in disjoint local regions of the network.

Table 2.6: Static MAS examples performing network management.

A particular case of Mobile Agents are the Swarm Intelligence systems. These are systems inspired by the biological behaviour of social insects (e.g. ants or bees). Swarm Intelligence is the property of a system whereby the collective behaviours of (unsophisticated, simple) agents interacting locally with their environment cause coherent functional global patterns to emerge [Bonabeau et al., 1999]. A well-studied case is the system based on the behaviour of ants when, for instance, they look for food. A classical application of these Ant Systems is the routing in telecommunication networks. Examples of Ant Systems are [Di Caro and Dorigo 1997], [White et al. 1997] and [Schoonderwoerd et al. 1996].

JAMES is an example of a Mobile Agent Platform for the Management of Telecommunication and Data Networks [Silva et al., 1999] aimed at the development an infrastructure of Mobile Agents with enhanced support for network management and trying to exploit the use of these technologies in some telecommunications software products. The platform was developed completely from scratch, using JAVA on behalf of the Eureka project 1921. The agents are started at the JAMES Manager and migrate through the network nodes and finally go back to the Manager. Each Mobile Agent has a specific Itinerary with a set of tasks (Missions) to be executed across the JAMES Agencies. To facilitate a more efficient migration of the agents, a flexible distribution of the code has been implemented. Fault-tolerance is specifically supported in JAMES. It includes a checkpoint-and-restart mechanism (which works by saving the checkpoint in persistent storage), a failure-detection scheme, an atomic migration protocol and some support for fault-management.

Finally, the integration of SNMP support in the JAMES platform is worth mentioning. SNMP services are provided by JAMES including services for interoperability with Legacy SNMP Managers.

Other Mobile Agent approaches for network management are summarised in Table 2.7.

References	*Description*
[Halls and Rooney, 1998]	ATM switch control, connection admission control, similar to MbD
[Gavalas et al., 2000]	Mobile agents – SNMP integration, SNMP table views, polling and filtering.
[White et al., 1998a] [White et al., 1998b]	Network modelling, fault management, Mobile Agent framework and architecture.
[Caripe et al., 1998]	Network-awareness applications, network monitoring
[Sahai and Morin, 1999]	Mobile Agent framework targeted to mobile user applications where users are intermittently connected to the network through unreliable or expensive connections.
[Bohoris et al., 2000]	Dynamic service management and reconfiguration, focusing on service performance and fault tolerance management, among others.
[Du et al., 2003]	Framework for Mobile Agent-based distributed network management with a high integration with SNMP.

Table 2.7: Examples of Mobile Agents in Telecommunication Networks.

All these examples show that Multi-Agent Systems are a suitable paradigm for implementing new network management tools aimed at improving network functionality.

Chapter 3 Network Resource Management

3.1 Overview

In Chapter 2, we presented the background to network management, network technologies and Software Agents. This chapter focuses on network resource management and specifically using the virtual network concept. First of all, we describe the virtual network concept and its main characteristics, then present several mechanisms for resource management using virtual network. We also present the pre-planned restoration mechanisms, which also utilise the logical network. Finally, the complexity of these tasks and selecting the use of Software Agents to perform these tasks are discussed. We then describe the desired properties and characteristics for a network resource management system and several metrics are reviewed and defined.

3.2 Logical Network Concept and Network Resource Management

A Logical or Virtual Network constitutes a higher layer, which is logically independent of the underlying physical network. Thus the topology resulting from the assignment of a set of Virtual Paths (in ATM) or Label Switched Paths (in MPLS/GMPLS) can be visualised as a virtual topology (Figure 3.1). We refer to these VPs or LSPs as Logical Paths (LPs). A set of resources can be allocated to each Logical Path, which includes bandwidth, although they can also be established without any resource reservation. This Logical Network acts as a physical network where the user connections or flows can be established. However, the main characteristic of this Logical Network is its flexibility; this means that it is possible to change the virtual topology, the capacities assigned to each LP, etc. as required and hence the logical network can be adapted to changes in traffic profiles [Sato et al., 1990] [Dziong et al., 1996a] [Leon-Garcia and Mason, 2003].

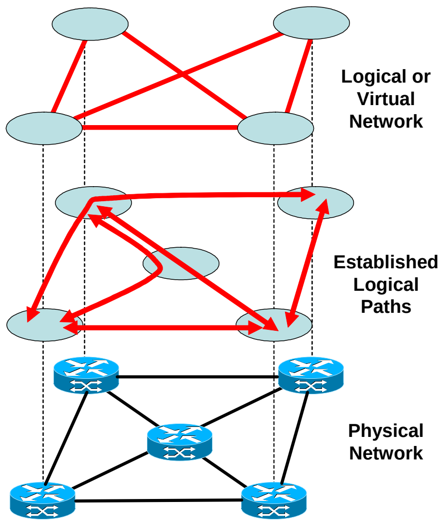


Fig. 3.1: Logical or Virtual Network Concept.

There are two main motivations for using a Logical Network. The first is to separate the management functions in order to customise them to the particular needs of certain services and user groups. The second is the virtual division of the network resources in order to simplify the resource management functions and provide QoS and Grade of Service (GoS) guarantees for certain services and user groups. GoS indicates how good the availability of a particular service is. Usually, the main component is the connection rejection probability. In general, the applications of LPs can be divided into three classes:

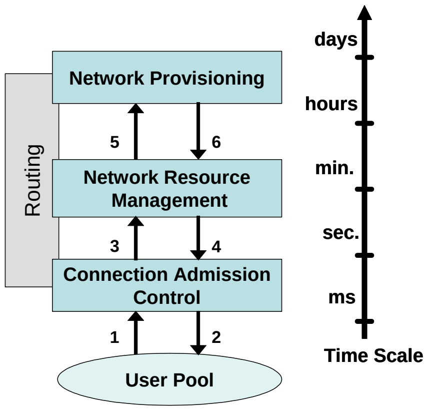
- Service oriented LPs, which are created to separate management functions specialised for different services and/or to simplify the QoS management (different LPs for different services). If there is allocation of resources this can also be used to provide sufficient GoS and fairness for different services. Moreover, bandwidth allocation also simplifies the bandwidth allocation to connections due to the homogeneity of the traffic of each LP.
- User oriented LPs, which are created for group users who have specific requirements (e.g. customised control algorithms, bandwidth under the user control, increased security and reliability, etc). Typically this is used to create VPNs.
- Management oriented LPs, which are created to facilitate some of the management functions not associated with any particular service or user group. Particularly interesting is the use of backup LPs in order to facilitate fault restoration. Also, the use of end-to-end LPs facilitate the connection establishment process for two reasons: the routing is already performed and the CAC decision can be based on local information only in the ingress node (there is no need for bandwidth reservation on transit nodes).

The main drawback of using LPs, in the case of bandwidth allocation for each LP, is that this could limit the possible statistical multiplexing gain. In other words, in networks with a greater number of origin/destination nodes, and/or fewer physical links with little capacity, there could be a significant number of LPs going through certain links that are dividing their capacity into small blocks of bandwidth.

Given a physical topology, the first assignment of LPs is the Logical Network design. The objective of a LP design is to optimise its topology and bandwidth allocation according to the predicted traffic demand and performance constraints. An optimisation algorithm is usually utilised for this purpose, e.g. [Dziong et al., 1996b] and [Qureshi and Harms, 1997]. This procedure is beyond the scope of this work. Once the Logical Paths are set up, the initial design may not be optimal, due to either discrepancies between the real and predicted traffic patterns or changes in them. Thus, an adaptive procedure is required to adjust the Logical Network bandwidth allocations and topology. This procedure should be based on measurements of parameters relevant to GoS, fairness, resource cost, etc. At present, this adaptation is usually performed periodically (e.g. every hour, morning / afternoon / night topologies, daily, weekly, etc) on a semi-permanent basis. This is usually done by the management centre because this adaptation typically consists of recalculating the whole Logical Network topology.

These processes are very similar, in terms of their objectives and use of LPs, to the Internet Traffic Engineering proposals [Awduche et al., 2002]. Traffic Engineering can be defined as an iterative process of network planning and network optimisation with the purpose of optimising resource efficiency and network performance of operational IP networks. Enhancing the performance of an operational network, at both the traffic and resource levels, are major objectives of Internet Traffic Engineering, along with facilitating reliable network operations. MPLS / GMPLS is a very important technology for Internet Traffic Engineering [Awduche et al., 1999] [Xiao et al., 2000] because it supports explicit LSPs, which allow constraint-based routing to be implemented efficiently in IP networks. Of course, these concepts can be used for Traffic Engineering in any kind of network, even outside Internet.

Network Resource Management includes, among others, the Logical Network management processes. Network Resource Management is a short-to-mid time set of functions related to the optimisation of the Logical Network and resource allocation based on the performance measures from the connection layer (Figure 3.2).



1) Connection demand
2) Connection accepted or rejected
3) Network performance problems
4) Virtual Network reconfiguration
5) Detected physical network bottlenecks
6) Physical network upgraded

Fig. 3.2: Network Resource Management Interrelations with other Management Mechanisms.

The Network Resource management layer is above the connection layer whose main function is concerned with the connection admission control and establishment of new user connections into the LPs. Particularly important is the logical allocation of bandwidth to connections in the connection layer. The importance of this allocation lies in the fact that it allows the separation of the traffic control algorithms on the connection and higher layers from the cell layer (which deals with resources at the cell level like buffers, scheduling policies, etc). This feature can simplify significantly the resource management and traffic control functions. Moreover, the bandwidth allocation to the user connections or flows makes the system similar to the circuit-switched environment on the connection layer. However, the bandwidth allocation to the user connections or flows is very complex and beyond the scope of this work.

On the other hand, Network Resource management is below the Network Provisioning layer, which acts in the long term. The main objective of Network Provisioning is the updating of the physical network topology based on the performance measures of the Logical Network, detecting physical bottlenecks, such as single-point, potentially dangerous nodes and links in the case of failure, etc. These functions analyse the network and propose physical network upgrades and improvements to the network provider.

All of these layer functions are closely related with the routing mechanisms. User connections need to be routed over the Logical Network and the LPs need to be routed over the physical network. The physical network improvements may significantly affect the LPs and connections routing. Therefore, routing has to be taken into account on every layer. In fact, Traffic Engineering mechanisms in MPLS / GMPLS are largely based on the utilisation of constraint-based routing [Ashwood-Smith and Berger, 2003] [Berger et al., 2003b] [Jamoussi et al., 2002] [Awduche et al., 2002] [Xiao et al., 2000].

Focussing only on the Logical Network management and the mechanisms that involve LP manipulation, the following sections introduce the main mechanisms for network resource management and fault protection. All of these mechanisms are part of the network resource management layer and are performed over the connection layer using the abstraction of having assigned bandwidth to the connections.

3.3 LP Bandwidth Management

Bandwidth management attempts to manage the bandwidth assigned to the LPs, i.e. changing their bandwidth in order to better adapt the Logical network to the offered traffic. The ultimate objective is to maximise network performance. When, due to unforeseen changes in the offered load and/or when the Logical Network design is not optimal, some LPs can become under-utilised and others congested, i.e. all the LP bandwidth is already assigned to several connections or flows and new connections on this LP are being rejected. These rejected connections could be accepted if the traffic loads were better balanced.

One of the main objectives of bandwidth management is to minimise the Connection Blocking Probability (CBP), i.e. the probability that an offered call is rejected due to insufficient capacity being available for the allocation of the new call. Minimising CBP can also be seen as maximising GoS for a particular service in the case where the LP carries only traffic of a single service.

There are two actions usually performed by the bandwidth management systems in order to increase the bandwidth of a congested LP: re-allocation of bandwidth and re-routing LPs [Friesen et al., 1996]. For instance, consider the situation shown in Figure 3.3, where there are three LPs beginning in node 1 and LP 1-2 is congested (i.e. is full and new connections from node 1 to node 2 are being rejected). Therefore LP 1-2 needs an increase in bandwidth.

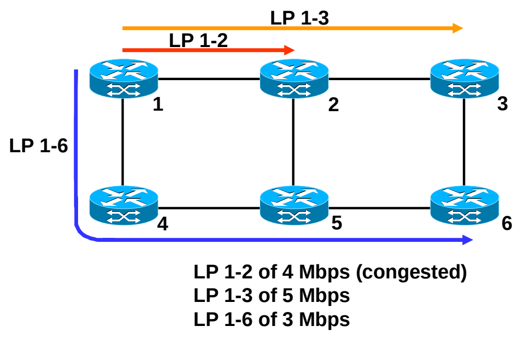


Fig. 3.3: Initial Situation. LP 1-2 is Congested.

Bandwidth re-allocation means that LP 1-2 bandwidth can be increased without the need for re-routing. The required bandwidth for increasing LP 1-2 can come from two possible sources: the available bandwidth of the physical links, not assigned to any other purpose; and the unutilised bandwidth already assigned to other LPs with a low load (pre-emption). Let us suppose that in the example, there is available bandwidth in the physical link 1-2, therefore this bandwidth can be assigned to the LP 1-2. If this is not enough and LP 1-2 is still congested, then it would be possible to reduce the bandwidth assigned to LP 1-3 and assign it to LP 1-2 (only if LP 1-3 has unutilised bandwidth or it has a lower priority than LP 1-2 and pre-emption is allowed).

Suppose that bandwidth re-allocation cannot be applied because one or more physical links in the LP's same physical path have no available bandwidth and bandwidth from other LPs sharing the same physical path cannot be utilised either (there is not enough bandwidth or pre-emption is not allowed). In such a case, the only possibility is to re-route LPs through other physical paths in order to have enough available bandwidth and then increase bandwidth to the congested LP. There are also two possibilities: re-route the congested path through a physical path with enough available bandwidth to meet its needs; or, re-route other LPs going through the same physical path as the congested LP in order to release bandwidth and make room to increase it to the congested LP.

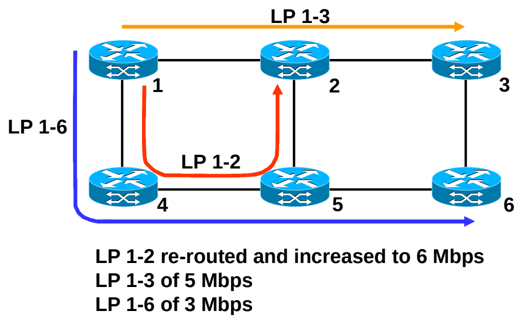


Fig. 3.4: The Congested LP 1-2 is Re-routed and its Bandwidth Increased.

Figure 3.4 shows the example where LP 1-2 is re-routed through another physical path with more available resources where it can be increased to 6 Mbps. Figure 3.5 shows the example where LP 1-3 is re-routed in order to make room in the physical link 1-2. Then the congested LP 1-2 can be increased.

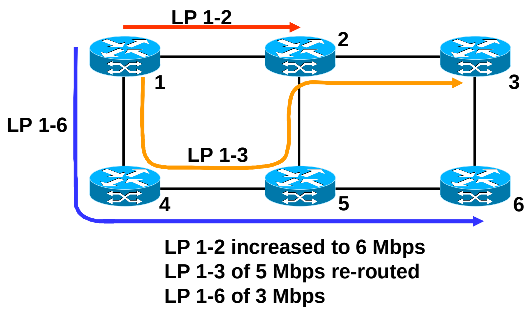


Fig. 3.5: The LP 1-3 is Re-routed and the Congested LP 1-2 can be Increased.

Re-allocation is preferable to re-routing because it is less traumatic for the already established connections going through the affected LP. Another possibility in re-routing is to have a transient situation where both the old and the new substitute LP coexist for some time. New connections are made in the new LP and the old one endures while it still has live connections. However, this option requires more bandwidth.

3.4 Fault Protection

One important aspect of network management is network survivability. The network must be capable of maintaining service continuity in the presence of faults. This objective can be achieved at a device level by developing more reliable network elements but this does not override the need for failure detection and restoration capabilities to be implemented at different network layers. The ultimate goal of the protection mechanisms is that customers do not notice failures.

Re-routing is traditionally used at the IP layer to restore service after link and node outages. Re-routing at the IP layer needs a period of routing convergence which may range from seconds to a few minutes and which, in many cases, could be too long. Faster restoration mechanisms can be used in the Logical Network layer in ATM / MPLS / GMPLS networks. Fast restoration is achieved by using pre-planned mechanisms, i.e. using backup LPs [Yahara and Kawamura, 1997] [Kawamura and Ohta, 1999] [Sharma et al. 2002]. A working LP can hence be protected using a backup LP (Figure 3.6). As soon as a fault affecting a working LP is detected, the connections going through it are rapidly switched over the backup LP with a minimum packet loss. The backup LP is already pre-established and does not need online calculation or convergence time. For instance, in Figure 3.6, a fault in the physical link 4-5 affects the working LP 1-5. This LP 1-5 is protected by the backup LP 1-5 and the connections going through the working LP are diverted to the backup LP. There are several backup techniques and each one has its advantages and drawbacks [Calle et al., 2001] [Marzo et al. 2003b].

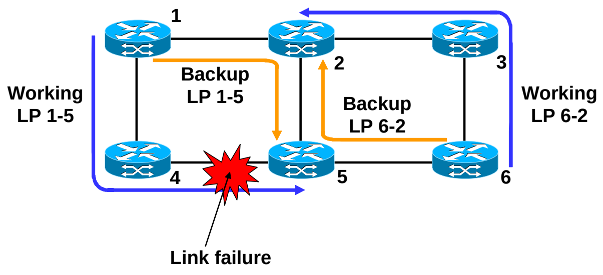


Fig. 3.6: Working and Backup LPs.

Protecting all the working LPs of the Logical Network with backup LPs could be very costly in terms of bandwidth consumption. Therefore, typically, only LPs carrying high priority traffic and/or LPs for special users/services are protected using backup LPs. Other LPs carrying low priority traffic may not be protected in that way, if at all. It is possible to visualise this situation as if it were the logical network divided into several different logical sub-networks, each one with different priority concerning its protection (e.g. a high priority network protected with backup LPs and a low priority network not protected at all). Of course, there could be more than two levels with different degrees of protection.

The establishment of backup LPs is usually done using similar procedures to that of the working LPs (i.e. using an optimisation procedure for Logical Network design). Constraint-based routing and explicit routing can also be used. In any case, a backup LP is closely related to the protected working LP and also has to be managed like any other LP (taking into account their special relationship).

3.5 Spare Capacity Management

Network providers want high revenues. Since bandwidth is an expensive resource, the objective is to minimise the bandwidth reserved for restoration procedures [Xiong and Mason, 1999] [Oh et al., 2000] [Sohn and Sung, 2002]. Therefore, besides the possible decision to protect only the high priority LPs, there is another decision to make: whether to protect the LPs against a single, simultaneous failure or to protect them against multiple simultaneous failures. This decision may depend on the failure statistics collected during the network operation and evaluating the probability of having multiple simultaneous failures on the network. This probability is typically very low and the network provider has to make a decision. This decision may also include information on whether the considered failures would be link failures or node failures. Node failures are more severe and can affect several links simultaneously.

In any case, after the decision is made, it may be possible to apply the mechanism of sharing the bandwidth reserved for several backup LPs in order to minimise it. For instance, suppose that what is required is protection against single link failures in the network shown in Figure 3.6. The backup LP 1-5 and the backup LP 6-2 both go through physical link 2-5. There is no need to reserve additional bandwidth for both backup LPs in that link, only the one that requires more. Using this technique means that none of the backup LPs have their required bandwidth already assigned, but rather, that the required bandwidth will be in a reserved pool for the backup path that really needs it when a failure occurs.

To control and manage this spare capacity in the links is another function directly related to the LPs and backup LPs management. Specifically, when protected LPs and their corresponding backup LPs are established, the required bandwidth has to be available on the links where there are backup LPs.

3.6 Resource Management Complexity

The three management functions presented in this chapter (LP bandwidth management, fault restoration and spare capacity management) are closely related to each other. This is because all of them deal with LP bandwidth. Instead of performing these functions separately, using independent mechanisms (performance management and fault management in the FCAPS model), an interesting way forward could be to group all these functions together in an LP management proposal.

In the case where bandwidth re-allocation (increasing or decreasing) is performed on an LP, then if this LP is protected with a backup LP, the amount of spare bandwidth must also be checked and perhaps increased or decreased as well. In the case of LP re-routing, it is possible that the backup LP also has to be re-routed and this would also affect the spare bandwidth in the links. This entire procedure has to be taken into account when load balancing is performed.

This is already quite a complex task but, in addition, the abnormal situations in the course of faults also have to be handled. When a fault occurs, the traffic on the affected LPs is automatically switched over to the backup LPs. While this situation continues, the spare bandwidth will be less than what is required. It could also be the case that a bandwidth adjustment was required while the traffic is going through the backup LP. Finally, when the failed components are restored, it would be of interest to return to the pre-fault situation of the LPs. Therefore Logical Network management during the course of faults turns out to be a special case.

On the other hand, these management functions must also deal with special configuration cases, such as network provider restrictions and constraints (LPs with and without pre-emption, with different levels of protection, with minimum and / or maximum bandwidth values, etc), Virtual Private Networks, LPs that cannot be re-routed, etc. In conclusion, carrying out such dynamic management in the context of a Logical Network – including all these functions – would be a very complex task.

3.6.1 Automatic Reconfiguration

User demand, in terms of the number of users and also the traffic each user causes, is growing faster than the network capacities and speeds. Moreover, the continuous introduction of new services and applications mean that traffic patterns may present fluctuations that make traffic load forecasting difficult. In order to get the maximum profit from their networks, network providers always want to have optimal or near optimal Logical Network design. Nowadays they usually rely on manual reconfigurations from the human network administrators, or periodic reconfigurations. This is usually performed at the management centre and these reconfigurations can be based on the use of traffic statistics and optimisation algorithms.

The perfect load balancing would be done by calculating the optimal Logical Network (working LPs, backup LPs and spare capacities) and updating it for every new established connection. This is not possible for two main reasons. First, it is not possible to calculate the optimal Logical Network fast enough since, to do that, a centralised algorithm is needed. This centralised algorithm would need an up-to-date network status which could be only maintained with a very low connection arrival rate. Secondly, there would be such a number of bandwidth re-adjustments, LP re-routings and update monitoring, that a huge amount of management traffic would be generated reducing, rather than increasing, network performance. This problem is described in [Aneroussis, 1996] for ATM and, to avoid this problem, a compromise must be found between the number of readjustments of the Logical Network and the distance to the optimal configuration.

For these reasons, among others, automatic, periodic adjustments (hourly / daily / weekly) are usually used. The management system uses an optimisation algorithm to calculate the optimal Logical Network based on traffic statistics collected between one bandwidth adjustment and the next. These periodic adjustments, however, have a drawback in that they cannot cope with short, unpredictable changes in the offered traffic load. There may be some situations where the network performance could be significantly reduced, e.g. when moments after the reconfiguration there is an unforeseen change in the offered load and the network will not be readjusted until the end of the current period.

The impossibility of dealing with unforeseen traffic loads means that there is a need for automatic reconfiguration when a problem arises. However, automatic reconfiguration of the Logical Network must be done carefully as mentioned previously. There is a need to minimise the topology changes and thus a balance must be found between the number of topology changes and the optimal level of the Logical Network.

3.7 Desired Characteristics of Joint Dynamic Bandwidth Management and Restoration of Logical Paths

The functions that an architecture for joint dynamic bandwidth management and restoration of LPs must support are as follows:

- **load balancing**, including re-allocation of bandwidth and re-routing of LPs. The load balancing mechanism is based on the minimisation of the CBP for each LP. This must be performed without overwhelming the network with excessive reconfigurations;
- **LP protection using backup LPs**, considering only the use of global backup LPs, i.e. there is only one backup LP for each working LP that should be protected. The working and backup LPs go through completely disjoint paths;
- **spare capacity management**, enabling the option of shared bandwidth with a protection level against one simultaneous failure of a single link

The desired characteristics of a system that integrates all the above defined management functions are as follows:

- **Integration**: the integration and co-ordination of the different mechanisms that make use of the logical paths with the objective of avoiding interference. Basically, when a reconfiguration occurs due to load balancing, the designed backups and placed spare capacities have to be also taken into account and updated if necessary. Also, when a fault occurs, the load balancing mechanisms must take it into account;
- **Modularity**: a virtual path management architecture should be modular as a whole, i.e. its behaviour must be like an independent management module with regard to the other control and management mechanisms in the network. This implies that this module can be activated or deactivated at the behest of the network provider and that it can be deployed only over certain sub-networks or network domains and not in others;
- **Robustness**: a virtual path management architecture must be robust in the way that if part of the system crashes it does not take down the whole system. Also, even if the entire system goes down, the network must continue operting – without the functions performed by the proposed architecture;
- **Scalability**: a suitable scalability level is desired, i.e. when the network grows the architecture must not degrade its operation and overwhelm the network with management messages;
- **Independence**: the joint dynamic bandwidth management and restoration module should not interfere nor substitute other management and control mechanisms in order to perform its function. That is, other mechanisms (e.g., CAC, connection routing, etc.) not dealing with the management of the existing LPs, must not notice the module actions;
- **Simplicity / Flexibility**: the module should be simple and flexible in the sense, on the one hand, that it must enable system updating and modification and on the other hand, it must not be too complex and represent too much of load for the network elements being managed.

3.7.1 Scalability

Scalability is a well-studied aspect in the fields of parallel computing and parallel programming. It is also studied from the perspective of distributed systems, e.g. client-server, but these studies usually refer to generic distributed systems or to particular services such as the Web. However, Network Management systems are different from generic distributed systems and they have special characteristics, for instance they should operate continuously, with high reliability and over several different time scales. Most scalability evaluation methods are based on the instruction count of every function in the system and/or on measures and tests performed in real system implementations. Thus, such evaluation includes both system design and system implementation. We believe that the overall system scalability depends mainly on the system design and architecture, while the implementation details only allow performance tuning and final adjustments.

Finding a scalability metric and comparing different systems is very difficult. In the literature, different scalability definitions are presented:

- [Jogalekar and Woodside, 1998] assert that 'a system is scalable if it can be deployed effectively and economically over a range of different sizes, that are suitably defined'. They define a metric based on the relation between the rate of providing valuable services, the quality of those services and the cost of providing those services.
- Another definition is proposed in [Burness et al., 1999] where scalability is considered as a measure of how the performance changes as the system size is increased. The performance is viewed as the response time, as seen by a user under normal working conditions, coupled with the cost of the system – hardware requirements – per user. The authors relate the scalability study with the system design, analysing several characteristics without giving an overall scalability measure. It is also pointed out that network communication is often the key performance limitation.
- A different definition of scalability is given in [Jardin, 1996] where it is defined as "the ability to augment the scope of the system (new functions, new classes, new instances). Network size, configuration and technologies keep changing. Management systems must be able to track and support this evolution in an incremental way without disrupting the service".
- Other research, such as [Lee et al. 1998], define scalability for MAS as "the average measure of the degree of performance degradation of individual agents in the society as their environmental loading, caused by an expansion in the size of the society, increases".

Thus it can be asserted that there is no clear definition of scalability and even less agreement on a method or measure to evaluate it. As a more generic definition that comprises the above definitions it is possible to say that a "network management system is scalable if its performance does not degrade as the system to be managed grows".

The scalability problems can arise for different causes or a combination of more than one. These causes can be classified in terms of Processing, Data storage and Communication problems. Processing problems arise when there is insufficient capability to execute the required processes in a node. This includes the lack of memory as well as the lack of computation capacity. Data storage problems arise when a large amount of data has to be stored in a node. Communications problems arise when management messages flood the network. If these three values are maintained under a defined threshold while the network size and number of users increase, then the system is scalable.

In [Jogalekar and Woodside, 1998] and [Jogalekar and Woodside, 2000], a scalability metric for distributed systems is presented. This metric encompasses logical and physical system aspects (it is based on a ratio between the system throughput and a combination of response time and cost) and is used for the comparison of two system configurations. The ratio of the two measures indicates whether the system is scalable or not.

It is necessary to evaluate the system scalability in several different sizes, for instance configuration 1 and configuration 2 can be compared using the following ratio, called p-scalability:

$$\Psi_P = \frac{F(\lambda_2, QoS_2, C_2)}{F(\lambda_1, QoS_1, C_1)}$$

(Formula 3.1)

where λ is the throughput, C is the cost and QoS represents a set of parameters that measures the quality of service perceived by users. F is a function which evaluates the performance and the economy of the system. It seems important that F should be proportional to λ/C. For instance, the QoS could be defined as a function of the mean delay *T* and the target delay *T'* as follows:

$$QoS = f(T/T') = \frac{1}{(1+\frac{T}{T'})}$$

(Formula 3.2)

Then manipulating the p-scalability definition, it can be defined as:

$$\Psi_P = \frac{\lambda_2 \cdot C_1 \cdot (T_1 + T')}{\lambda_1 \cdot C_2 \cdot (T_2 + T')}$$

(Formula 3.3)

When all scaling variables can be expressed as a function of a parameter *k*, the scaling of the system can be defined by this scaling factor *k*. The following Figure 3.7 shows different scalability situations as a function of this scaling factor.

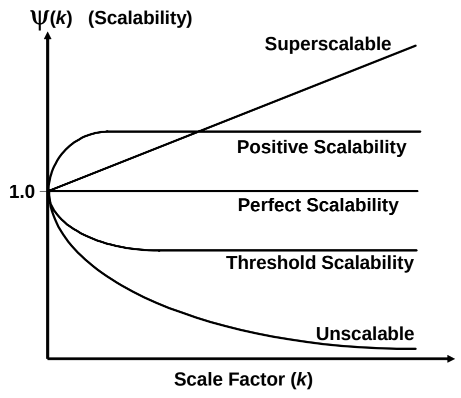


Fig. 3.7: Different Scalability Behaviours.

This scalability metric cannot be used for direct comparison of different systems. However it is suitable for evaluating the scalability of each system and asserting how well it scales.

3.7.2 Modularity

Modularity can be defined from two different points of view. First of all, it can be defined by taking into account the relationship between different components or modules of the same network management system. Secondly, it can also be defined by considering the relationship of the system with other network management systems that perform the same functions.

Regarding the relationship between different components or modules of the same network management system, a possible definition is that a system can be considered to be modular if it is clearly divided in modules or building blocks that implement individual or small groups of functions. An interesting property of these modules is that they can be switched on or off as needed. If the module is switched off (or it crashes) the functions performed by this module simply stop without disrupting the network services and connections already established.

Regarding the relationship of the system with other network management systems that perform the same functions, a good modular system is one that can be utilised on a sub-network or a network domain without influencing the rest of the network or other neighbouring network domains that are utilising different modules to perform the same functions or even not performing these functions at all.

Many network management system approaches based on MAS propose the substitution of the conventional mechanisms for the new ones. In this case, the new approach must implement not only the new functions, but also the functions already performed by the previous management system. This obviously increases the complexity of the approach and such integration could become dangerous because when the system goes down, all the services could be disrupted. This is the case for instance of the Tele-MACS approach [Hayzelden, 1998] [Hayzelden and Bigham, 1998], where all the functions depend partially or totally on a centralised agent (the Tactical Survival Agent) and the Hybrid approach [Somers, 1996] [Somers et al., 1997], which requires a significant number of complex deliberative agents.

Modularity is also a good property in terms of faster system design and implementation and enabling a simple way of continuously improving and updating of the modules.

3.7.3 Robustness

Robustness can be defined as the ability of a system to overcome failures, crashes, lack of information, disruptions, etc. i.e. unforeseen problems, with a minimum loss of effectiveness. Robustness can also be seen from two different points of view. First of all, when a hardware or software component failure affects our system behaviour and this component is not part of our system. In other words, all the components of our system are working properly, but an external component failure is affecting our system operation. For instance, a typical situation is when all the components of a distributed system are working properly and a lack of communication (e.g. due to a failed link) mean that some of the system components become isolated from the rest of the system. This is usually dealt with by using robust communication protocols that can manage communication disruption situations. The second situation is when the hardware or software component that fails is part of our system. In this case, such a situation may not affect the rest of the components or cause them to crash. The failed component must be restarted as quickly as possible, ensuring that the situation before the failure is restored. Typically, there may be information losses, bad configurations, etc. and the rest of the components must have mechanisms in order to help the restored component to return to a proper working situation.

Modularity could be of great help in these cases, because perhaps only one or a small number of modules within the overall system would eventually fail or crash and the system may simply continue its basic operation with several non-critical functions disabled.

Typically the best and most commonly-used mechanism for ensuring system robustness is replication. Replication means that everything or at least the crucial components and data of the system are replicated, i.e. there are copies of these components and data in another location. In the case of a failure occurring, these backup components simply substitute the original ones and system operations resume. This does, of course, require doubling or almost doubling the resources used, which means a significant increase in costs. Moreover new problems may arise, e.g. maintaining data integrity could be difficult in a network management system where there is continuous monitoring from the network.

There are two ways of synchronising the system in charge and the backup system. The first option is to establish some kind of periodic synchronisation and if the system in charge fails, the backup system resumes the operation from the last synchronisation point. This technique is not useful in network management because crucial information would be out of date (e.g. connections established and released after the last synchronisation point). The second option is to have the system in charge and the backup system operating exactly equal, as if they were both in charge. This could mean doubling the monitoring and just letting the commands of one system reach the network elements. This kind of protection will work for network management systems.

These mechanisms for providing a system with robustness could easily be implemented in a centralised management system (only the central manager will be replicated). However, these mechanisms are very difficult to implement in a distributed system. A distributed system must base its robustness on the fact that if some of its components fail, the remainder continue in charge and when these components are restored then they must be brought up-to-date, as fast as possible, by the ones that did not fail.

Software Agents, a natural means of deploying distributed systems, also have the ability to work with a lack of information and they can easily implement mechanisms that collaborate in restoring a failed agent.

Chapter 4 Dynamic Virtual Path Management Architecture based on Software Agents

4.1 Overview

This chapter details the proposed architecture, the different types of Software Agents and their functions, the cooperation mechanisms, and the selected decision making mechanisms. It also details the decision making process, and the communication features between agents.

Firstly, the overall architecture is presented in section 4.3, and then, the different types of agents are described. Section 4.4 describes the different algorithms proposed for network monitoring, bandwidth reallocation, logical path rerouting, the utilised restoration scheme, and the spare bandwidth management. Finally, section 4.5 describes the integration of all these mechanisms.

The proposed architecture should fulfil the desired properties and objectives proposed in the previous chapter, basically that is scalability, robustness, modularity, and simplicity.

4.2 Introduction

There are proposals to perform dynamic load balancing, and there are also proposals to perform fault management based on backup paths. It is possible to find centralised and distributed algorithms to perform these tasks separately. However, there are no proposals that integrate both functions together and, as it is illustrated in the previous chapter, there is a need for these functions to act in a co-ordinated way. Moreover, conventionally distributed algorithms usually have difficulties in handling exceptions, special configurations, and constraints. Also, centralised algorithms have the drawback of their lack of scalability and the possibly slow response time.

Therefore, a possible way of obtaining such co-ordination is to integrate both load balancing and fault management functions in the same management system. Performing both these functions involves a high degree of complexity. Using MAS in order to cope with these complex distributed tasks has been found to be useful. On the other hand MAS can also be seen as a design metaphor, i.e. Software Agents provide designers and developers with a way of structuring an application around autonomous, communicative elements, which helps in the development of complex systems, especially in the open and dynamic environments [Jennings, 2001].

The aim of this thesis is to propose a dynamic logical path management architecture for a network resource management module in order to cope with the management functions that utilise the Logical Network, i.e. that deals with logical paths and resources. The main resource to manage is the bandwidth or capacity of the transmission links. These functions are load balancing, fault protection management using backup paths, and the corresponding spare capacity management for a certain network protection level. These functions have already been analysed in a previous chapter.

The complexity of the proposed functions, the desired objectives (integration, modularity, robustness, scalability, independency, simplicity and flexibility), and the trends in network management lead us to propose the use of Software Agents in a completely distributed architecture detailed in this chapter. Due to these complex functions must be performed in a core network scenario with high availability and capacities, the use of Mobile Agents is not recommended. This is because the system would not be able to take advantage of the special characteristics of the Mobile Agents and the added complexity of the Mobile Agent systems to already complex tasks would not be worthwhile. Therefore, the architecture based on MAS should be composed of fixed or static agents only.

The desired objective of achieving good scalability suggests the use of a completely distributed management architecture without a global network view. In this case, it inevitably means that an optimal Logical Network configuration cannot be guaranteed. However, the proposed joint dynamic bandwidth management and restoration system based on MAS could start its operation with an already established Logical Network, which can be optimally designed by another network planning system. Starting from this point the system works making the necessary minimal readjustments in order to adapt the logical topology when a problem arises, i.e. when an LP becomes congested and new connections are being rejected.

The proposed joint dynamic bandwidth management and restoration can be seen as a module, and it could be integrated with the conventional network resource management mechanisms (see Figure 4.1 in comparison with Figure 3.2), with the main goal of automating some of the day-to-day tasks of the human network administrators. The module must act in a transparent way for the other layer mechanisms, specifically the Admission Control and Routing of the user connections or flows.

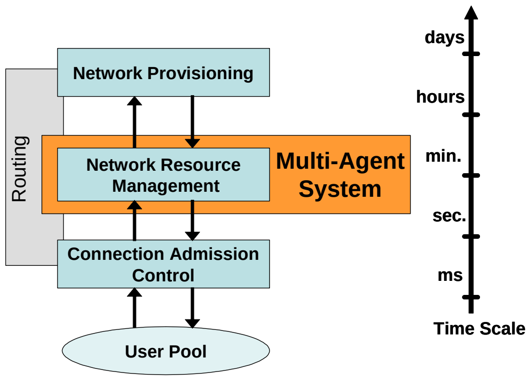


Fig. 4.1: Multi-Agent System Relation with the Network Resource Management Mechanisms.

4.3 Proposed Architecture

The proposed architecture follows a distributed scheme without a management centre. This means that there is no central network view where optimisation algorithms can be applied. Nor is it the case of maintaining several distributed views of the whole network due to scalability reasons, and also data synchronisation / integrity problems.

Two types of agents are proposed: the network Monitoring (M) agents and the Performance (P) agents. All these agents are situated at the different network nodes where the computing facilities are. There is one P agent and several M agents on every node as displayed in Figure 4.2. M agents are subordinate to P agents, therefore the proposed architecture can be seen as a two-level agent hierarchy. Each agent (M or P) is responsible for monitoring and controlling the resources they are assigned to and react quickly when an event occurs.

In particular, M agents are responsible for the following tasks:

- Monitoring a single logical path from its origin and detecting congestion.
- Maintaining the reference to the backup path assigned to the logical path it is responsible for.
- Performing the switchover function from the working logical path to the backup path when receiving the failure alarm (from the P agent).

It is better (faster) that the node control system implements the switchover mechanism, in this case the M agent just notices that there has been a failure and the backup path has been activated, and acts accordingly. However if the node control system does not implement that mechanism, the M agents can also implement it and change, when the failure has been communicated, the connections from the working logical paths affected by the failure to their backup paths.

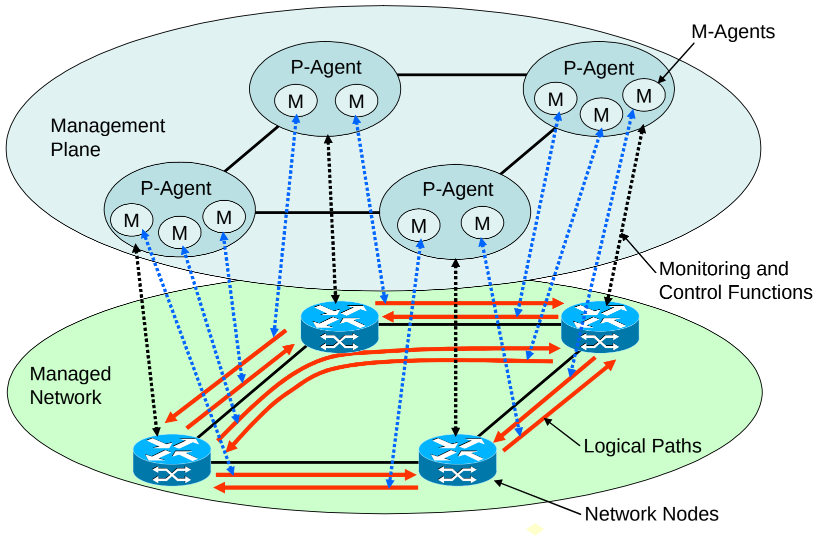


Fig. 4.2: Performance and Monitoring Agent Distribution.

Therefore there is one M agent per unidirectional working logical path. If the logical paths are established and released by another system or manually by a human network manager the M agents must be created or destroyed accordingly by the P agents. Note that the proposed architecture acts over an already established logical network and it is supposed that the number of logical paths and the set of source-destination pairs remain equal.

A P agent centralises the monitoring and control tasks of a whole node, also including the physical links. Each P agent maintains node status information and keeps track of the established logical paths starting at or going through the node. For every physical link, the status of the established logical paths and the bandwidth assigned to them, the spare bandwidth reserved for the backup paths, the backup paths' configuration, the free bandwidth on the link, etc. are maintained. They are responsible for receiving the failure alarms from other P agents and also from the node control system monitored by the P agent. The main functions of the P agents are:

- Monitoring nodes and physical links, and receiving the fault alarms from the node control system and from other P agents.
- Maintenance information of a partial logical network status.

- Management (create, destroy, modify, update, consult, etc.) of the M agents in the same node. The creation/destruction of M agents is carried out when another management system establishes/releases logical paths.
- Collaboration and communications with other P agents in order to achieve their goals.

P agents are responsible for the communications with the node control system. All the monitoring tasks performed by M agents and the P agents must be done through previous establishment of communications with the node control system. If the software agents are placed in the same node, these communications can be established by means of an Application Programming Interface (API). There are also other possibilities, for instance monitoring and controlling the node through a SNMP agent, which would enable remote monitoring.

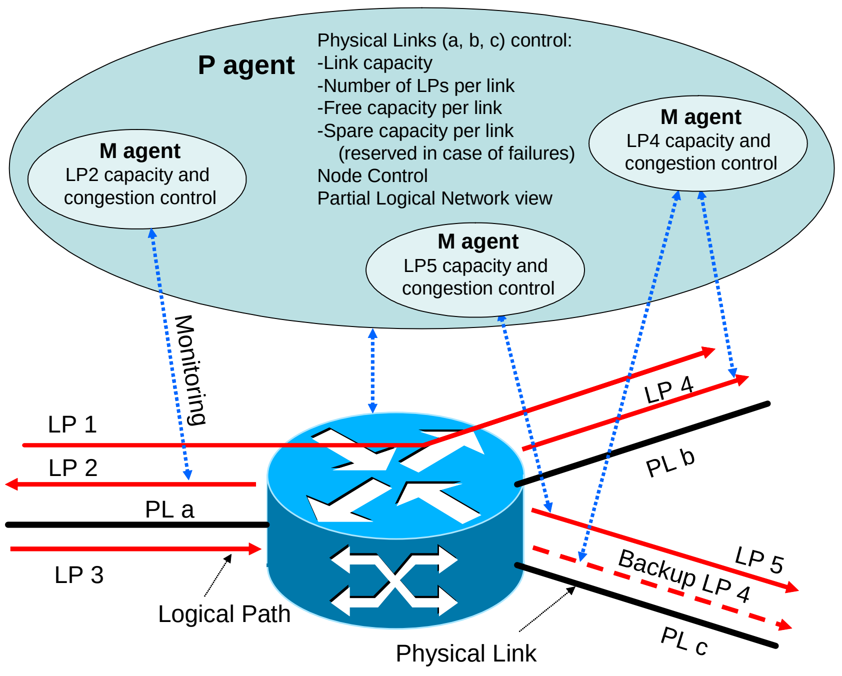


Fig. 4.3: Detail of the Agents Placed in one Node.

Figure 4.3 shows a network node being managed by one P agent and three M agents. Note that there is an M agent for each logical path starting in the node and there are not any M agents for the logical paths ending or going through that node. Note also that if a logical path is protected by a backup path both are the responsibility of the same M agent, which also could perform the switchover. Figure 4.3 summarises the main tasks for each agent.

4.3.1 Agent Distribution

One of the main characteristics of distributed architectures is the proximity of the decision making to the managed elements of the network. Hence the distribution of the processing load becomes more balanced since it is not concentrated in a single point. Another advantage of a distributed system is its robustness against failures.

Therefore the agents of our proposed architecture can be placed in the same nodes and communicate directly with the node control system by means of an API (Figure 4.4a). If the agents are placed in the same node it is also possible that an SNMP agent in the node would allow the agents to monitor and control the node through that SNMP agent (Figure 4.4b).

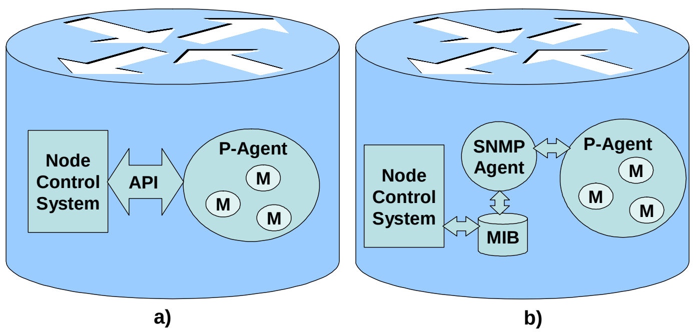


Fig. 4.4: Agents and Node Control System Communications. Agents in the same Node Cases.

Using SNMP agents in the managed nodes, it is also possible to establish an independent management network and place the management agents in different nodes other than the managed ones. This is particularly interesting in optical networks, with the option of having a dedicated non-optical network for management purposes only. This network can be seen as a control plane (Figure 4.5).

Sometimes the transport nodes have limitations in their processing capabilities or in their available memory / storage capacities. This separation between the transport network and the control network may help to alleviate the transport nodes' load on the one hand and, on the other hand to have powerful computers dedicated to the control and management tasks. Moreover, the communications at the control network can also be focussed on management purposes, without interfering user connections. However, this control network also has to be managed.

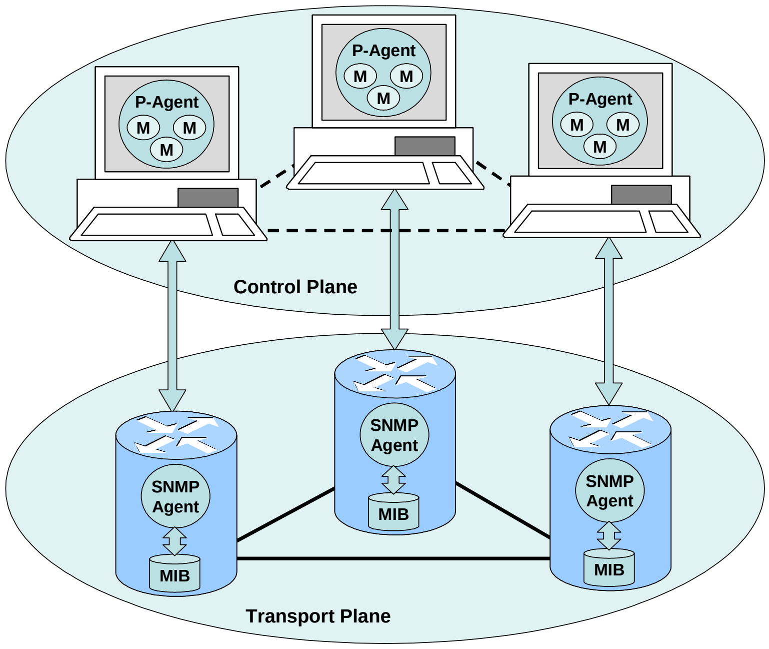


Fig. 4.5: Agents and Node Control System Communications. Agents in different Node Case.

The proposed architecture can perform on-line management by integrating the software agents in the same nodes that are being managed, and it can also perform off-line management by establishing an independent control network.

At this point, a nomenclature is introduced in order to facilitate the organisation of the examples where there are several P agents and several M agents. The nodes are numerated using positive integer values {1, 2, 3, …} and the P agents named as Pa1, Pa2, Pa3, etc. according to the node number where the P agent resides; in a similar way to the nodes, all the LPs in the network are labelled as LP1, LP2, LP3, etc., and consequently the monitoring M agents are Ma1, Ma2, Ma3, etc. The LPs are usually depicted as arrows in the Figures to clearly distinguish the LP direction. However, we have called the physical links Link 1-2 or Link 5-4 making reference to the number of the origin and destination nodes (e.g. the physical link from node 1 to node 2 or the physical link from node 5 to node 4).

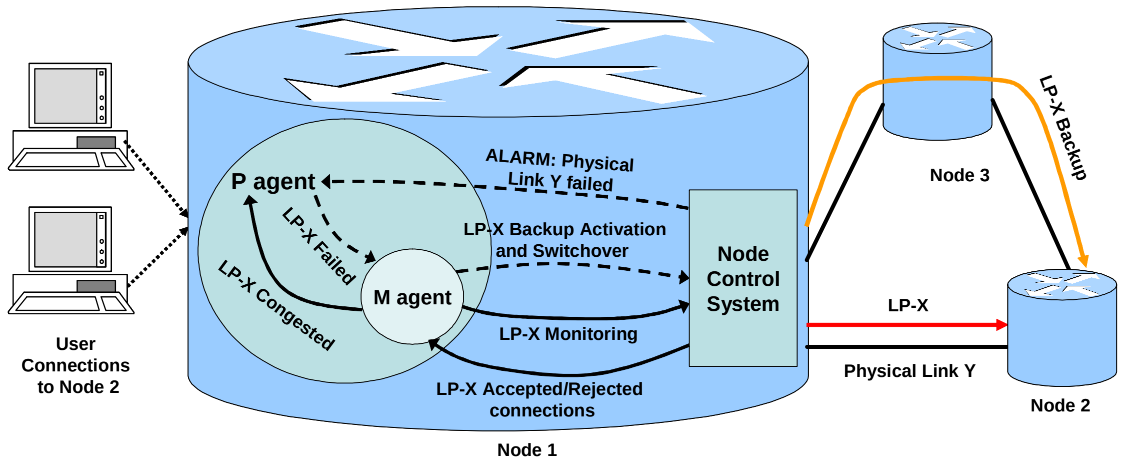
4.3.2 Monitoring Agents (M Agents)

Monitoring agents are simple reactive agents (lightweight processes) with a stimulus / response type of behaviour, whose main mission is to monitor and control the assigned resource (a logical path).

An M agent periodically monitors a single logical path from its origin. The input data from the node control system is the number of offered and the number of rejected (or accepted) user connections for this logical path. The main task of an M agent is to decide when the monitored logical path is congested. There are many ways to make such a decision, which can be from a simple threshold value to a complex deliberation considering many parameters. It is possible to utilise any of the artificial intelligence techniques for decision making, for instance Case Based Reasoning (CBR), Fuzzy Logic, etc. it is even possible to use Learning techniques that show the M agents how to detect that the logical path is congested.

However, those experiences using sophisticated methods are left as a future work following the research line of this work. We propose several simple methods, shown in section 4.4, which allow us to experiment with the monitoring period, the congestion detection mechanism, and the amount of capacity in which to increment a congested logical path.

Figure 4.6 shows the main functions and relations of the M agents. It is possible to group the M agents' rules/actions into three classes: Failure actions, Congestion actions, and Monitoring actions. These actions have assigned priorities according to the class they belong to. Therefore failure actions have the highest priority and are pre-emptive, i.e. they can interrupt other actions. Congestion actions also have priority over Monitoring actions.



User Connections to Node 2
P agent
M agent
LP-X Failed
LP-X Congested
ALARM: Physical Link Y failed
LP-X Backup Activation and Switchover
LP-X Monitoring
LP-X Accepted/Rejected connections
Node Control System
Node 1
Node 3
LP-X Backup
LP-X
Physical Link Y
Node 2

Fig. 4.6: The main Functions and Relations of M agents.

Failure actions are performed when a failed link in the network is detected. All the M agents that control a logical path affected by the link failure are notified. Congestion actions are not displayed in Figure 4.6 due to the fact that they are decisions made by the P agent and related to changing the logical paths' capacities and re-routing them. Monitoring actions are performed while M agents ask the Node Control System for some parameter values. The M agents' rules / actions have priority level, the ones related to failures have the highest priority, followed by the congestion ones, and finally the monitoring ones have the least priority.

M agents may gather some information due to their monitoring task. This information should be on the forms of knowledge and statistics rather than bulk data and it must be kept as low as possible. However, the size and type of this information depends on the method used for the selected decision making mechanism.

As said above, the main decision an M agent must make is to decide when the logical path being monitored is congested. However, there are several other decisions that an M agent can make or help to make. Basically these decisions are the monitoring period, and the recommended amount of capacity by which to increment a congested logical path. Both decisions can also be made by a P agent or they can even be fixed values manually configured by the human network administrator.

The monitoring period must not be too short which would cause an overload on the node control system, i.e. monitoring that was too fast could cause too much processing in the node control system. On the other hand, a monitoring period that is too long could mean that the agent does not find out about some congestion problems fast enough. Therefore there is a safety interval for the monitoring period. This safety interval must be selected depending on the network dynamics.

The amount of capacity to increment a congested logical path must also be within an interval. The minimum capacity to increment a logical path is the capacity that allows the admission control mechanism to accept one more connection in that particular logical path. Therefore this capacity corresponds to the largest possible connection type or class that this LP is accepting. The maximum capacity to increment a logical path will correspond to the smallest of the amounts of free capacities along the links that the LP traverses.

Note that in this work we consider that all the physical links and logical paths as unidirectional communication elements. If two nodes establish a bidirectional communication this means that two LPs must be set up, one in each direction. We consider each LP as independent, one from the other, hence they could have different capacities assigned to them for each direction and even follow different physical paths. If it is desired to force the LPs to have the same capacity in both directions and / or follow the same physical path, then there must be coordination between the two M agents in charge of these two LPs, one in each origin. Figure 4.7 shows this situation.

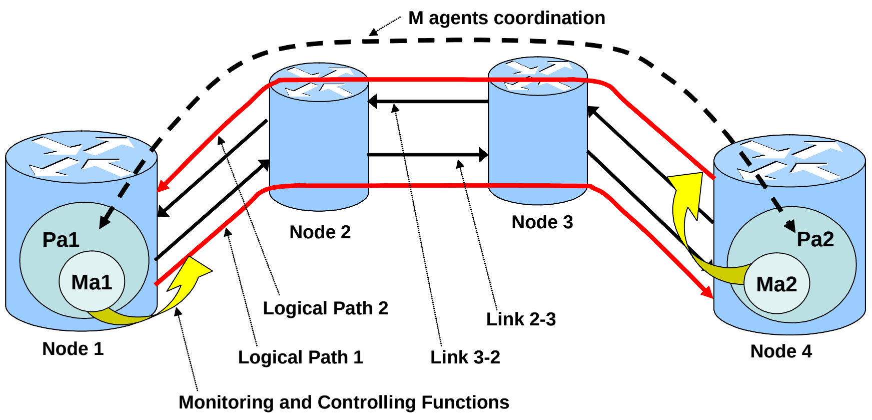


Fig. 4.7: Bidirectional Communications and M agents Co-ordination.

Therefore, when an M agent wants to perform a change in its LP configuration and this LP is tied to the LP in the opposite direction, the M agent must request its peer M agent to perform the same reconfiguration. This special case in M agent coordination between M agents that are in different nodes must be performed through the P agent, which is responsible for any communications outside the node.

4.3.3 Performance Agents (P Agents)

P agents are more complex than M agents. P agents perform several important tasks and may make several decisions by themselves or in collaboration with other P agents and M agents. When an M agent notifies the P agent that the LP is congested, then the P agent must decide what is the appropriate action to upturn the situation. Usually, the agent will check different possibilities and then decide what is the best action to take.

In our proposal, P agents base their decisions on a partial view of the Logical Network that each P agent maintains. Each P agent has a complete scheme of the physical network, and it is supposed that it doesn't change. If it could change it would only be necessary to update it for all the P agents. Following this scheme for the physical network, each P agent maintains two types of information from the Logical Network: a) the information the P agent is sure about, i.e. the information that is directly monitored by itself and its M agents, and b) the information the P agent is not sure about, i.e. information it has received from other P agents in the past and that could be out of date. This information is assigned a credibility degree, which is reduced according to the time that has passed. This information is updated asynchronously, only when there is a problem and P agents communicate with their neighbours, then the partial network view of an affected agent is sent attached with the coordination messages. This idea is displayed in Figure 4.8. We have chosen not to refresh this information regularly so as to avoid having too many coordination messages, which could affect the scalability.

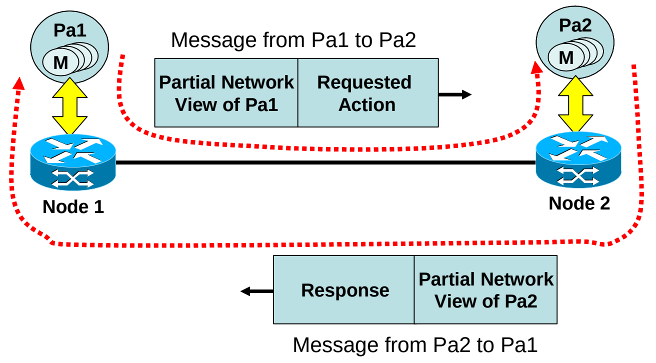


Fig 4.8: Partial Network Views attached to the P agents' Messages

This implies that the information in a P agent's partial network view (which tends to be really a whole network view with out of date information) must not represent a large amount of information. Thus this partial network view must only contain information strictly necessary for P agents, and must be coded in small size messages.

Let us suppose that there are no problems in the logical network and therefore P agents are not sending coordination messages. In such a situation there is no refresh from other P agents and the partial network view of each agent tend to be the information that each P agent monitors directly, since the other information is completely out of date and the P agent cannot rely on it. Suppose now that in this scenario an M agent detects a congested LP. The affected P agent may send a message to one of its neighbours requesting some action, the message has the P agent's partial view of the logical network attached to it. The P agent that receives the message firstly merges the partial network view of the P agent sender with its own partial network view and then it uses its actualised partial network view to make a decision or take action.

Imagine the following example: there is a physical network, depicted in Figure 4.9-a, where a full meshed Logical Network (20 unidirectional LPs from each node to all other nodes, not displayed in Figure 4.9) is established. In this case the partial view of the P agent at Node 1 (Pa1) and the partial view of the Pa5 could be the ones displayed in Figure 4.9-b, where the arrows represent the LPs a P agent is sure about and the dotted arrows represent information a P agent could have but it could be out of date. Note that Pa5 is aware of LP4 that starts at Node 1, going through Node 5. Although there is no M agent monitoring this LP at Node 5, the Pa5 receives all the requirements from Pa1 in order to perform readjustments on LP4. Therefore Pa5 is always up to date regarding LP4's status. Also note that P agents know the whole physical network but they can only be sure of the links they monitor directly.

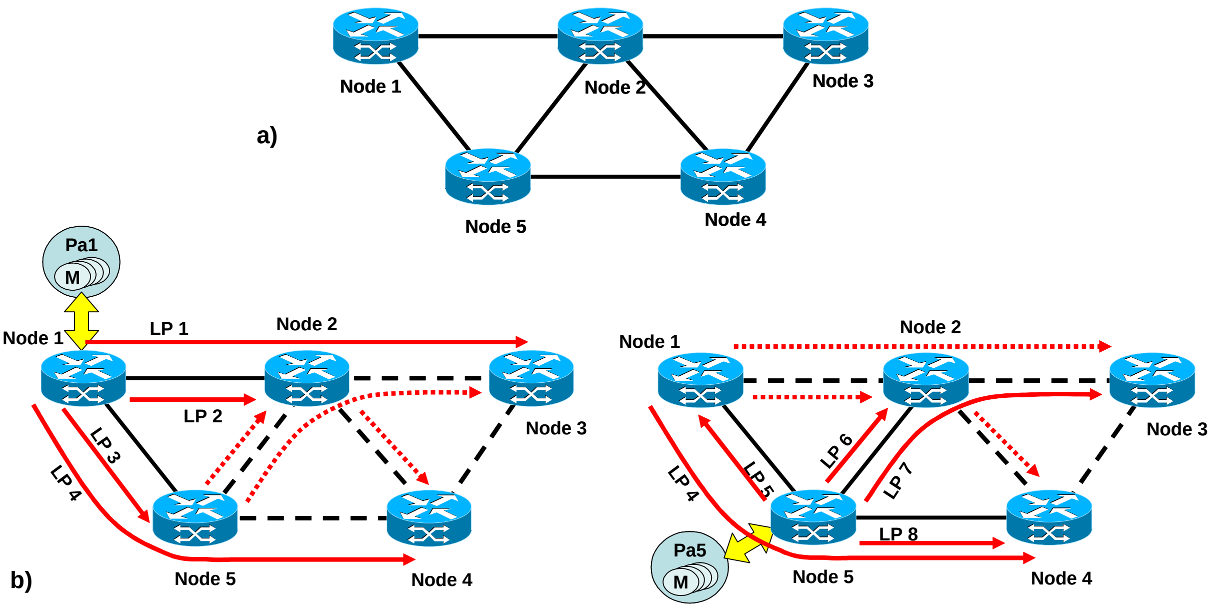


Fig. 4.9: Examples of Possible Partial Network Views of P agents at Node 1 and Node 5.

Therefore the agents must make their decisions with partial, incomplete and/or out-of-date information. The main goal of each P agent is to achieve the maximum utilisation of the resources it manages. This objective is equivalent to minimising the Connection Blocking Probability (CBP) for all the LPs that start in a particular node. In order to achieve their goal, P agents must cooperate with each other. Therefore, communication mechanisms and protocols must be used.

Regarding this communication between P agents there are several options. The use of proprietary protocols, the use of standard agent protocols (typically application level protocols with a rich capacity for transferring knowledge), or even it could be possible to make use of the Operations and Maintenance (OAM) -style of packets or cells by the definition of new types of these special packets or cells.

One of the scalability problems in distributed systems is produced when every agent in the system needs to communicate with all the other agents quite often. In order to assure good scalability we propose the constraint that every P agent can only communicate with its physical neighbours. Of course there are advantages and drawbacks of using this restriction. The advantages are that agents have a limited number of “known” physical neighbours and this simplifies the communication management between the P agents, and helps achieve good scalability. The drawbacks are that communication mechanisms are less flexible and, in most cases, slower due to the hop-by-hop message processing by the P agents. On the other hand, this restriction allows the possibility of establishing special LPs between physical neighbouring nodes for management purposes only. Therefore the communication protocols we propose, situated in an application level, have similarities with the signalling ones. The communication restriction along with the simplicity of the decision making mechanisms of our proposed agents mean that the communication richness in this case is not too high. Although it is perfectly possible to use standardised agent protocols, we have chosen to implement simpler communication protocols. Moreover, standardised agent protocols usually utilise a centralised name server, where all the agents have to be registered, and this would also have to be modified.

Again, the decision making mechanism could be very complex, deliberative, based on artificial intelligence techniques, etc., or it could be more simple. In any case the decision mechanism must be a distributed one. We have selected simple mechanisms based on heuristics, establishing an order of precedence between the possible actions. This precedence depends on the cost of each action, in terms of time and number of messages. Actions with a lower cost have more precedence than other actions with a higher cost. The P agent may not know when it is possible to complete a specific action depending on other P agents, their partial view of the network and their cooperation. The proposed mechanisms and algorithms are described in section 4.4. We leave the study of more complex mechanisms for future work.

The considered actions were presented in chapter 3 and correspond to the bandwidth reallocation and LP rerouting actions. For instance, consider the situation in Figure 4.9, and consider that an M agent at node 1 detects that LP4 is congested. In short, the different possible actions are:

a) assign free bandwidth from the physical links to LP-4. This is usually the less costly action because it only causes changes in LP-4.
b) try to reduce the capacity of other LPs and assign it to LP-4.
c) reroute one or more LPs in order to release capacities that can be assigned to LP-4 or reroute LP-4 itself through a physical path where there is enough free capacity to enlarge it.

Figure 4.10 shows a possible conversation between agents Pa1 and Pa5 in order to increase the bandwidth assigned to LP4. The case displayed in the Figure 4.10 represents that agent Pa1 detects enough free capacity in the physical link 1-2 and does not have enough information about the free capacity of link 5-4. Agent Pa1 decides to assign free capacity to increase LP4. The procedure is the following: it makes a pre-reservation in link 1-2 and sends a message to the Pa5 agent, which checks the available bandwidth in link 5-4. Supposing there is not enough free bandwidth in link 5-4, Pa5 could decide to try to reduce the bandwidth of LP8, and if this is possible then LP4 can be effectively increased. It could also be possible that Pa5 cannot reduce the bandwidth of LP8 (e.g. LP8 has a high level of connections at that moment or there are no pre-emption rights over LP8 and its unused bandwidth cannot be assigned to other LPs) then it would send back a negative message to Pa1 attaching the information of Pa5's partial network view. In that case, Pa1 updates its partial network view with the information from Pa5 and tries to find another possibility in order to increase LP4.

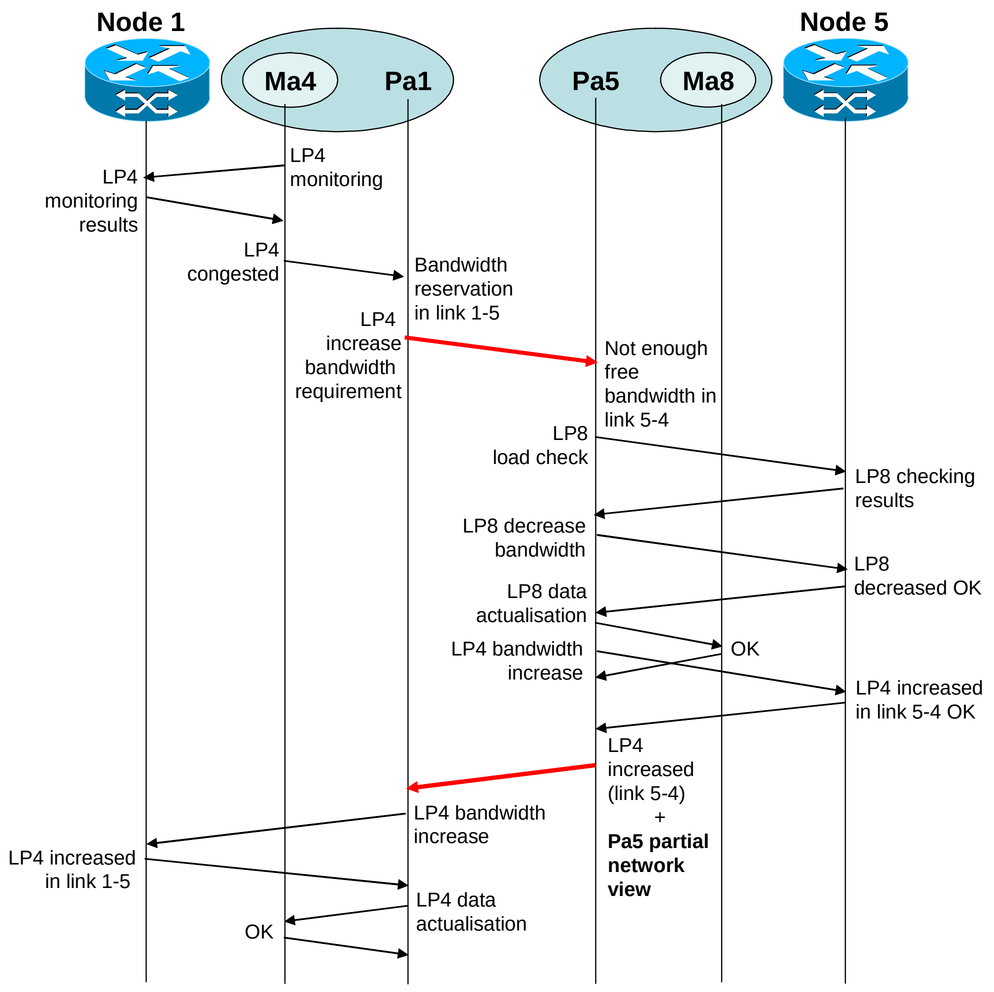


Fig. 4.10: Example of Communications between Agents and between Node Control Systems. (Red Arrows denote Inter-node Communications, while Black Arrows denote Intra-node Communications).

Note that P agents also manage the M agents. If another Network Management System or a Human Network Manager decides to establish or release LPs, then the P agents must be informed and they must create or destroy M agents accordingly.

The spare capacity in the links reserved for the backup paths along with the backup paths themselves, are managed in a similar way to the working LPs. The only difference is that the backup paths can be established without assigning any capacity to them (i.e. with zero bandwidth), and the required link capacity for the backup paths in case of failure is reserved. The technique of sharing reserved capacity for several different backups can be applied if the desired protection level is, for instance, against one single link failure.

4.3.4 Architecture Implementation Overview

This sub-section is introduced here in order to give an idea of how the proposed architecture has been implemented and tested, and to facilitate the understanding of the remainder of this chapter and the experiments presented in chapter 5.

In the version of the proposed architecture that has been implemented, we have used the Java language in an object oriented design paradigm. A P agent has been designed as a stand-alone program. Each P agent is capable of executing the required number of M agents as Java threads. There is no real communication between a P agent and its M agents, because they are part of the same program and share the same variables. The only real communications are between P agents. In this case we used the Java Remote Method Invocation (RMI) in order to implement the required messages and protocols.

The proposed architecture has been tested using a simulated network [Marzo et al., 2003a], which was also implemented as a distributed system. The simulator processes communicate with each other using the socket interface following a client/server model. The P agents' processes also act as client processes for the simulator node processes (which are the servers). The whole set of processes (simulator and management architecture) can be executed in a single computer or distributed in several computers. Figure 4.11 gives an idea of the simulation and the management architecture processes and their communications. More information about the simulation characteristics can be found in appendix A.

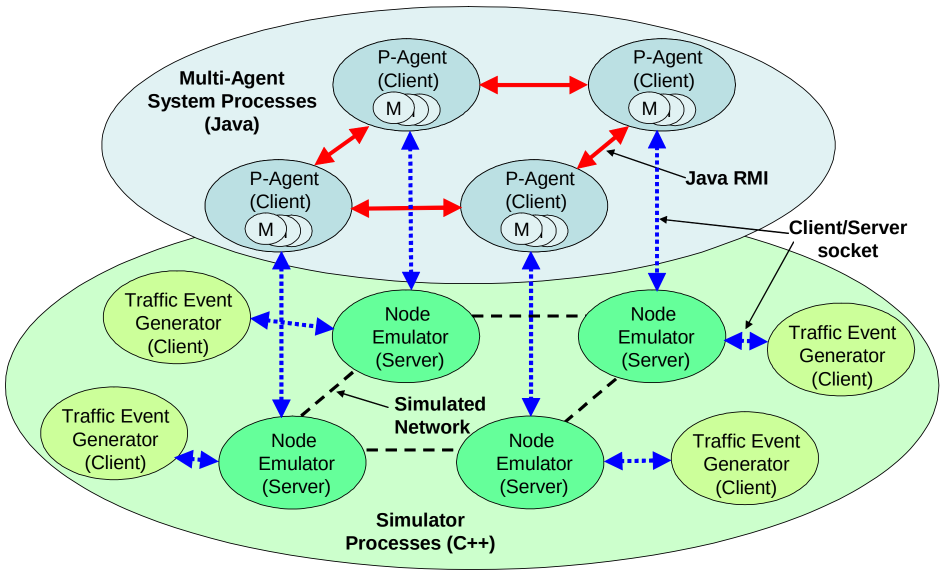


Fig. 4.11: Multi-Agent System and Network Simulator Processes and their Communications.

4.4 Proposed Algorithms

This section details the mechanisms (algorithms, protocols, heuristics, etc) that have been proposed as some of the many possible implementations of the agents' mechanisms. As stated before, the proposed mechanisms are simple, rule based, and use several heuristics. As one of the architecture objectives is simplicity, the proposed mechanisms could also be a good solution.

4.4.1 Monitoring and Congestion Detection

This sub-section presents the proposed mechanisms for the M agents in order to decide when an LP is congested. We have called this mechanism the Triggering Function (TF) and we propose 3 different TFs with the purpose of comparing them and viewing what their properties are. We can define a TF as a function that receives the monitored parameter values and returns true if the LP is considered congested or false if otherwise. Every time an M agent monitors its assigned LP the Triggering Function is executed. If the LP is considered congested, an alarm message is sent to the corresponding P agent.

Applying again the objectives of simplicity and scalability, we propose lightweight monitoring of LPs. This lightweight monitoring is achieved in three different ways:

- M agents are not monitoring the LPs continuously, instead there is a monitoring period of seconds or tens of seconds. M agent processes are usually halted and only resumed briefly every time the monitoring period expires.

- The simplicity of the TF, that it can be evaluated very quickly and is not resource consuming at all.
- The low number of monitored parameters.

We consider that a continuously changing logical network, with many management messages, could result in poor performance due to the management overload. The fast but not real-time monitoring combined with a tuned TF for the M agents is aimed at warning the P agents just when congestion is likely to happen.

In addition to the monitoring period and the selected TF, there is another factor that must be taken into account. When an M agent warns a P agent that a given LP is congested, then the P agent tries to increase the bandwidth assigned to that LP. Well, this increment of bandwidth must be of a given size. We have called this increment size the "step" size, and it also affects the number of times an LP is detected to be congested. For instance, if this step size is too small, it is likely that after the LP increment it could continue to be congested, and if the step size is too large then it is likely that the increased LP will not be congested for a long period. This second situation may provoke an LP capacity reduction. The possible combinations of these three factors are examined in chapter 5.

The M agents obtain the values of the monitored variables from the Node Control System directly or through an SNMP agent, as stated previously. In fact the monitored variables are closely related to the Admission Control mechanism. More specifically they are related to the number of accepted connections, the number of rejected connections, and the bandwidth already assigned to the currently established connections of a given LP. We assume that these variables are very common and almost all the Node Control Systems and/or the SNMP MIBs have them usually available. However, we think that it would not be very difficult to keep this information and make it available for management purposes, and it could be easily maintained by the Admission Control mechanism. The monitored variables are (for a given LP):

- The total number of offered connections (OC). This is a counter that keeps track of the number of offered connections from the start of the LP operation (e.g. $OC = 1233$).
- The total number of rejected connections (RC). This is a counter that keeps track of the number of the offered connections that have been rejected. Always $RC \leq OC$ (E.g. $RC = 125$).
- The current LP load (L). This is the amount of bandwidth assigned to the established connections of an LP just at that moment, given as a percentage of the total amount of bandwidth assigned to the LP.

Note that the offered/rejected information could also be obtained under other forms, such as accepted/rejected (e.g. 1108/125) connections or offered/accepted (e.g. 1233/1108) connections. We also assume that these variables are accumulative and their value never decreases in the LP life.

Note also that the LP load is independent of the real traffic load of the current established connections. As the values of the variables defined above are obtained periodically, we define them in terms of time. *OC(t)*, *RC(t)*, and *L(t)* are the variable values corresponding to the monitoring time *t*.

The three Triggering Functions are called *Rejected(t, limit)*, *CBP_{30}(t, limit)*, and *Load(t, limit)*. Their input parameter *limit* is the threshold value in order to consider the congested LP monitored . In the first case it is an absolute value and in the second and third cases it is a percentage. These functions depend on time (*t*) because they are evaluated periodically. All these functions have the same output: *one* if the LP is considered congested, and *zero* if otherwise. If the output is *1* then the M agent sends the corresponding alarm message to the P agent.

The idea of the *Rejected(t, limit)* function is to count the rejected connections in the successive monitoring periods, i.e. successive values of *RC(t)*. If there are rejected connections in the present period (i.e. *RC(t) > RC(t-1)*) then the rejected connections in the present period (i.e. the difference *RC(t) – RC(t-1)*) are accumulated in an internal counter of the M agent. If in the present period there are no rejected connections (i.e. *RC(t) = RC(t-1)*) then the M agent counter is reset to zero. When the value of the counter is equal or greater than the given *limit* then the LSP is considered congested. A formal definition now follows.

First of all a previous definition is needed:

$$count(t) = \begin{cases} count(t-1) + RC(t) - RC(t-1) & \text{if } RC(t) - RC(t-1) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Formula 4.1)}$$

Then the Triggering Function *Rejected(t, limit)* is defined as follows:

$$Rejected(t, limit) = \begin{cases} 1 & count(t) \geq limit \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Formula 4.2)}$$

The idea of this function is to allow a few occasional rejections, but if the rejections persist in the time and/or there are many rejections and the limit is exceeded then the LP is considered congested. Table 4.1 shows a numerical example:

T	*OC(t)*	*RC(t)*	*Count(t)*	*Rejected(t, 5)*
0	0	0	0	0
1	15	2	2	0
2	22	3	3	0
3	34	3	0	0
4	48	6	3	0
5	62	9	6	1

Table 4.1: *Rejected* Function Numerical Example

The idea of the *CBP*$_{30}$*(t, limit)* Triggering Function is to evaluate the Connection Blocking Probability (CBP) for the last 30 offered connections. There is a problem in calculating this CBP due to lack of information, as the only values obtained from the Node Control System are the *OC(t)*, *RC(t)*, and *L(t)* at a given time *t*. Using periodic monitoring as we have defined the distribution of the accepted and rejected connections are usually unknown . For this reason the *CBP*$_{30}$*(t, limit)* function calculates the CBP in the worst case, i.e. the case when the accepted connections are all grouped at the beginning of the monitored period and the rejections are all grouped at the end of the monitored period. Some previous definitions are also necessary. We have defined a sequence of bits a_n ,where *n* is an integer value, as follows:

$$a_i = \begin{cases} 0 & OC(t-1) < i < OC(t) - RC(t) \\ 1 & OC(t) - RC(t) \le i \le OC(t) \end{cases}$$

(Formula 4.3)

If there have been offered calls in a monitoring period (i.e. *OC(t)* > *OC(t-1)*) then *OC(t) – OC(t-1)* bits are added to this sequence, zeros for the amount of accepted connections in that period and ones for the amount of rejected connections, in this particular order. Then the *CBP*$_{30}$*(t, limit)* evaluates the last *30* elements of this sequence, and is defined as follows:

$$CBP_{30}(t, limit) = \begin{cases} 1 & \frac{1}{30} \sum_{i=OC(t)-29}^{OC(t)} a_i \ge limit \\ 0 & \text{otherwise} \end{cases}$$

(Formula 4.4)

This can be easily implemented using the idea of a shift register. It is necessary to consider that $a_{-29},..,a_0$ has been initialised to zero to operate properly. The greater the window size (*30* in this case) the longer the history taken into account for the CBP calculus. Consequently a big window size means that more connections have to be rejected in order to achieve the limit CBP. However a very small window size could give a meaningless result. Although the window size could be an input parameter (*CBP(t, limit, window)*) we fixed it at the value of *30* in order to have a certain amount of offered connections for the CBP calculus. Another reason was not to have too many open parameters for the experiments. Figure 4.12 shows a numerical example.

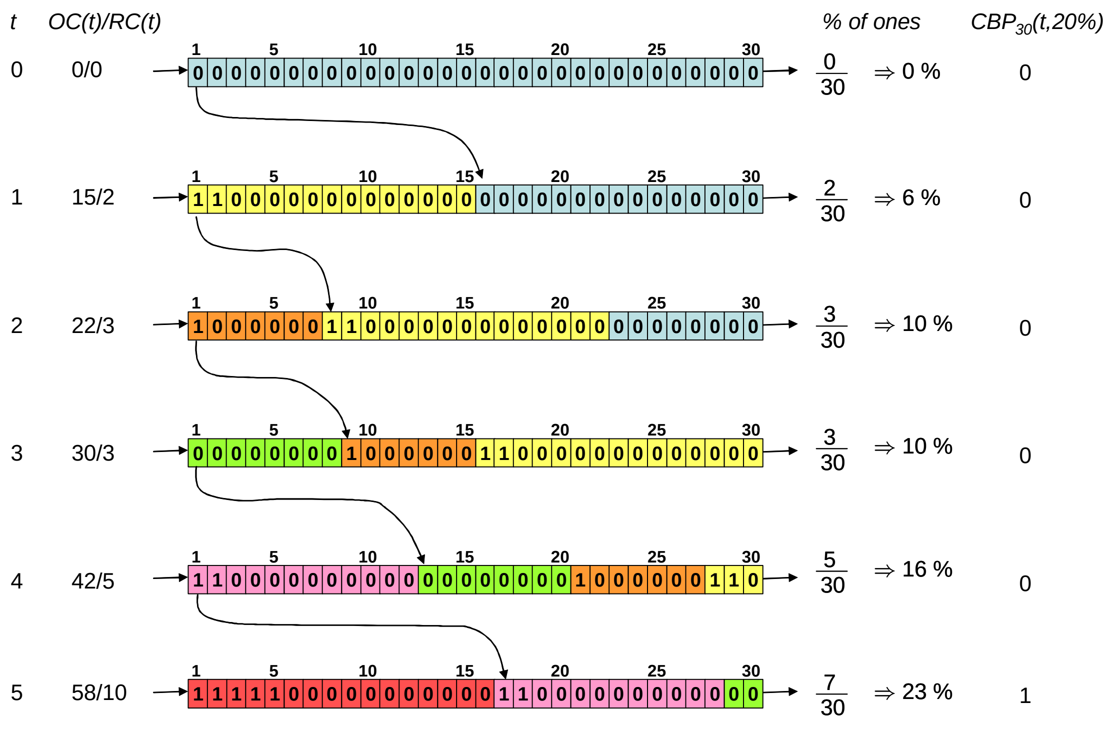


Fig. 4.12: *CBP_{30}(t, 20%)* Numerical Function Example.

The idea of this Triggering Function (*CBP_{30}*) is the same as the *Rejected* one, i.e. it allows a certain number of rejections, but if in the last *30* connection requests there are a lot of rejections and the limit value is exceeded then the LP is considered congested.

Both these first two TFs (*CBP_{30}* and *Rejected*) can be applied using any monitoring period. However if the monitoring period is too high compared with the amount of offered connection rate, then these functions may become meaningless. That is, the monitoring period should be defined in a way that the number of offered connections in each monitoring period is not greater than a given limit with a given probability. This supposes that the offered connection rate is known, which is usually not true.

The first two Triggering Functions are reactive, i.e. they wait until a certain level of rejection (given by the limit) is produced. The third function, *Load(t, limit)*, can be considered as a proactive one, because it tries to solve the problem before it exists. The idea of the *Load(t, limit)* function is as simple as if the percentage of occupation of the monitored LP exceeds the given limit then the alarm is sent in order to increase the LP capacity. Therefore, before the LP bandwidth is fully assigned to user connections and it begins rejecting, the P agent tries to increase it. In this case, the function is defined as follows:

$$Load(t, limit) = \begin{cases} 1 & L(t) \geq limit \\ 0 & \text{otherwise} \end{cases}$$

(Formula 4.5)

This is a very simple function, which, does not depend on past values, but rather just on the instantaneous load. Note that this load is the amount of LP bandwidth assigned to user connections given as a percentage of the total amount of LP bandwidth, and it does not reflect the real occupation of the user connections.

Although the same TF can be used in the whole network, the independence of the M agents makes the combination of all three defined TFs possible in the same network. This introduces a lot of flexibility, because each LP can be monitored in a different way. Besides, not only can it use a different TF, but it can also use different threshold limits, different step sizes, different monitoring periods, etc. for monitoring each LP. There can even be several LPs that can be fixed and thus not monitored at all. The use of different TFs and parameters for monitoring LPs depends on the type or class of the LP, its priority, or the characteristics of the user connections it transports.

The decision of which TF and which parameters to use can be left to the human network administrator, or it could also be a decision made by the agents themselves. It could be possible that the M agents could dynamically and automatically adapt their behaviour to the network status. This is beyond the scope of this work and hence included in the future work section as an interesting issue to investigate.

4.4.2 Bandwidth Reallocation

This sub-section describes the algorithms proposed for the P agents when reallocating the bandwidth assigned to the LPs. i.e. increasing or decreasing it. The algorithms presented in this section are designed to minimise the messages between P agents in order to achieve a better overall system scalability. First, a detailed description and illustrative examples are given in order to understand how the algorithms work. Secondly formalised definitions of the algorithms are presented.

In the case of a P agent receiving an alarm indication from an M agent that the LP it is monitoring is congested, the P agent's objective is to try to increase the bandwidth of the congested LP. Note that the congestion is always detected at the origin of the LP by the M agent responsible for it. There are several cases, depending on whether the LP spans over a single link or over several links, and whether there is free capacity in the links the LP traverse or not. If there is not enough free capacity to expand an LP, then there is the possibility of taking unused bandwidth already assigned to other LPs (pre-emption).

In order to try to achieve the P agent's objective of increasing the bandwidth of a congested LP three different algorithms are proposed. We call them *Free-Bandwidth-Only* (FBO), *First-Node-Only* (FNO), and *Any-LP* (ALP).

Free-Bandwidth-Only (FBO): In this case the congested LP can only be increased by means of the free bandwidth in the link. If the congested LP spans over several links there must be enough free bandwidth in all the links. Therefore the only task of the P agents is to check whether there is free bandwidth in the physical links the LP goes through. This task is very simple and does not require any coordination messages between the implied P agents. Therefore, the first P agent (at the origin of the LP) sends a message downstream asking for an LP to be increased and each P agent implied in the process makes a check. Figures 4.13 and 4.14 show this procedure. If there is enough free bandwidth it is pre-reserved and the message continues downstream to the next P agent. When the last P agent (Pa3 that manages the LP span from node 3 to node 4) is reached then the message is returned upstream with the confirmation and the new LP bandwidth is allocated. When the message arrives to the original P agent, then the LP capacity is effectively increased. Note that the *IncreaseBW* messages are acknowledged except in the last span, where they are implicitly acknowledged by the *Response* messages. The *Response* messages are also acknowledged. Henceforward the acknowledged messages will be omitted for better clarity in the figures.

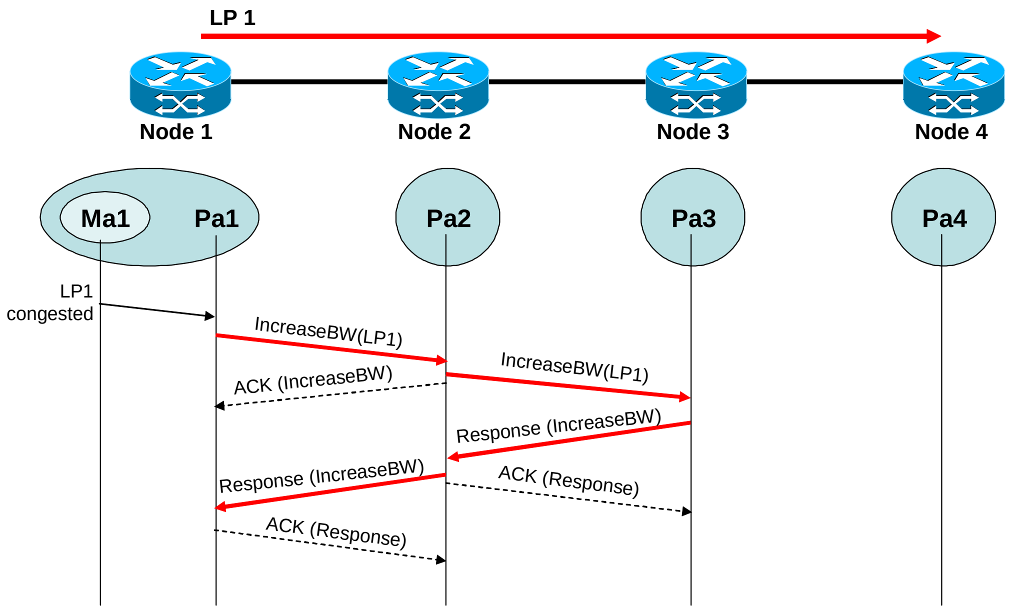


Fig. 4.13: Example of the Messages Sent for Increasing the LP1 Bandwidth.

In this example Pa1, following the FBO algorithm, checks if there is enough free bandwidth in link 1-2, and in that case it reserves part of this free bandwidth (Figure 4.14-A) and sends an *IncreaseBW* message to Pa2. When Pa2 receives this message it also checks the free bandwidth, in this case in link 2-3, and if there is enough free bandwidth Pa2 reserves the necessary amount (Figure 4.14-B) and sends an *IncreaseBW* message to Pa3. Pa3 performs the same actions in link 3-4 and as this is the last LP1 span then if there is enough bandwidth rather than reserving it Pa3 increases the LP1 directly (Figure 4.14-C). Then Pa3 sends a *Response* message upstream indicating the success or failure of the increasing bandwidth action. When Pa2 receives a positive *Response* message it increases LP1 in link 2-3 with the previously reserved bandwidth (Figure 4.14-D) and sends the *Response* message to Pa1. Finally Pa1 receives the positive *Response* message and increases the first LP1 span (Figure 4.14-E).

If Pa1, Pa2, or Pa3 detect that there is not enough free bandwidth in the physical links to increase LP1 then a negative *Response* message is sent upstream and the reserved bandwidth becomes available again. Using the FBO algorithm there are no other possibilities to increase the LP1 bandwidth in the case that there is not enough free bandwidth in one or more of the physical links it traverses.

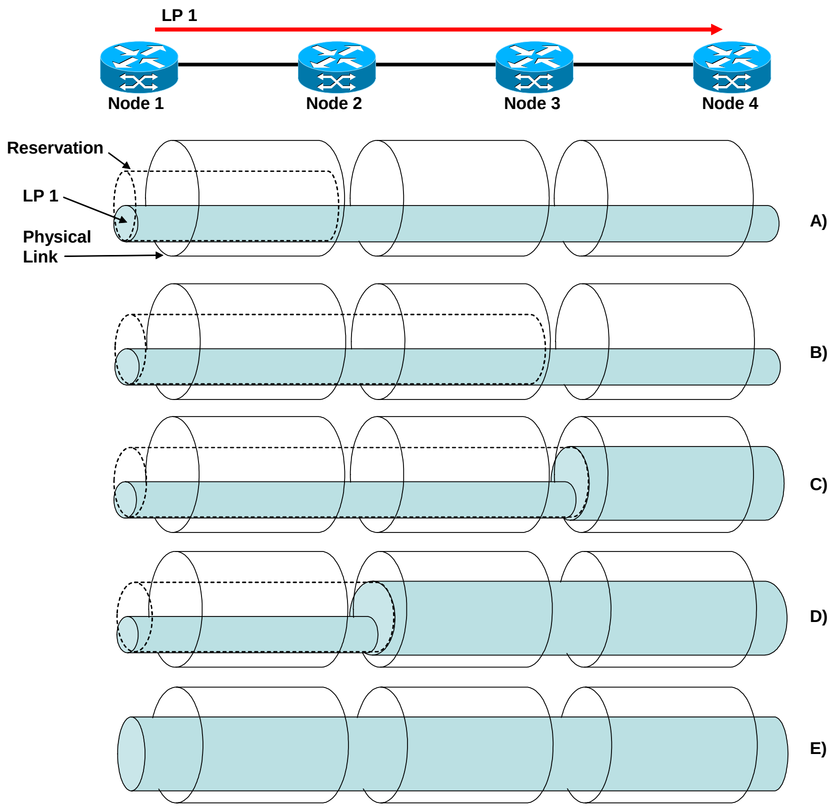


Fig. 4.14: Detail of the Mechanism for Increasing the Bandwidth of an LP.

There is an important issue with this algorithm. Given a Logical Network, the FBO algorithm is capable of adapting the capacities of the LPs, only increasing them when congestion is detected. After some time of using this algorithm, it is foreseeable that several LPs have been increased and there is not enough free bandwidth left to make other LP increases possible. That is, this algorithm is capable of adapting the Logical Network until a certain point and then the Logical Network becomes static and no more adaptations can be produced. Therefore it is necessary to introduce another mechanism that periodically (over a longer term) looks for LPs with a high percentage of underused bandwidth with the purpose of reducing their bandwidth and therefore having free bandwidth again. This will then enable a continuous adaptation of the Logical Network by using the FBO algorithm.

First-Node-Only (FNO): This algorithm is a minor modification of the previous one, the FBO. In this case, in the node origin of the congested LP the free bandwidth in the link is checked, and in the case that there is enough free bandwidth to enlarge the congested LP, the FNO's behaviour is exactly the same as the FBO algorithm. In the case that there is not enough free bandwidth in the first link, it is possible to look for other LPs and check if one or more of them are being underused in order to reduce their bandwidth, and assign it to the congested LP (pre-emption). In this algorithm, the LP candidates must begin in that same node and go through the same first physical link. Imagine the situation depicted in Figure 4.15 where LP1 is congested, the Pa1 using the FNO algorithm initially checks if there is enough free bandwidth in link 1-2. In the case that there is not enough free bandwidth, then LPs 2 and 3 can be considered with the possibility of reducing their bandwidths if they are not using them.

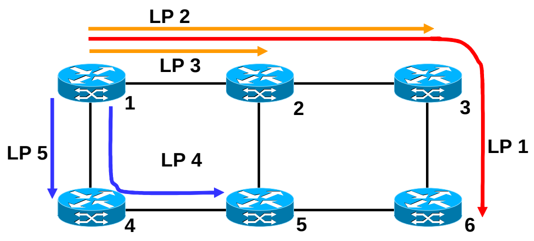


Fig. 4.15: Example of Candidate LPs to be Reduced (LP2, and LP3).
LPs 4 and 5 cannot be Candidates.

Supposing it is possible to reduce LP2's capacity, then a mechanism for reducing the bandwidth assigned to an LP is also necessary. In order to minimise the number of messages sent between P agents, we constrained the number of LPs to reduce their bandwidth to one. This implies that only one of the possible LP candidates to be reduced can be effectively reduced and only in the case that it has a greater amount of unused bandwidth than the amount required to increase the congested LP. Therefore, it is not possible to reduce small amounts of bandwidth from several candidate LPs in order to obtain the required amount for the congested LP.

The mechanism for reducing the bandwidth of an LP is depicted in Figures 4.16 and 4.17. The P agent at the origin node reduces the assigned bandwidth of the first LP span and sends a *ReduceBW* message downstream. Every P agent acts in the same way until the P agent managing the last LP span, which reduces the bandwidth and sends back a *Response* message confirming the reduction.

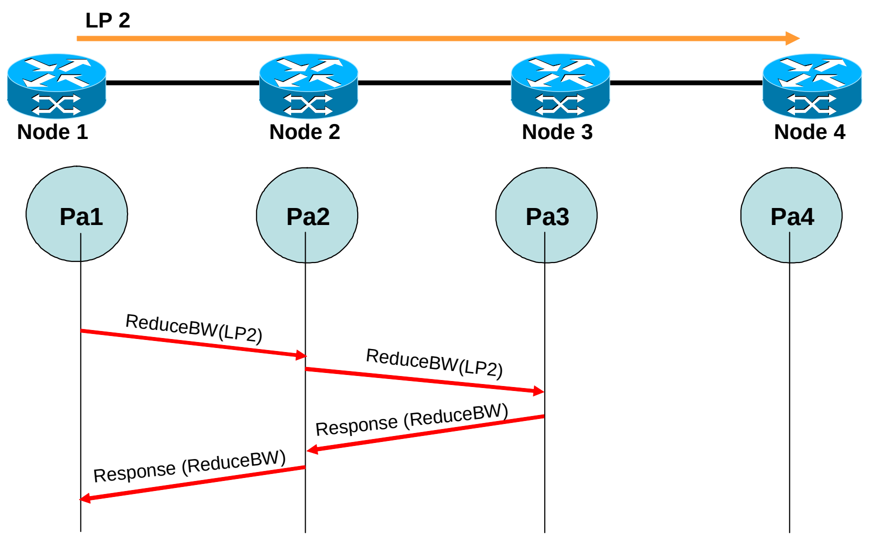


Fig. 4.16: Example of the Messages Sent for Reducing the LP2 Bandwidth.

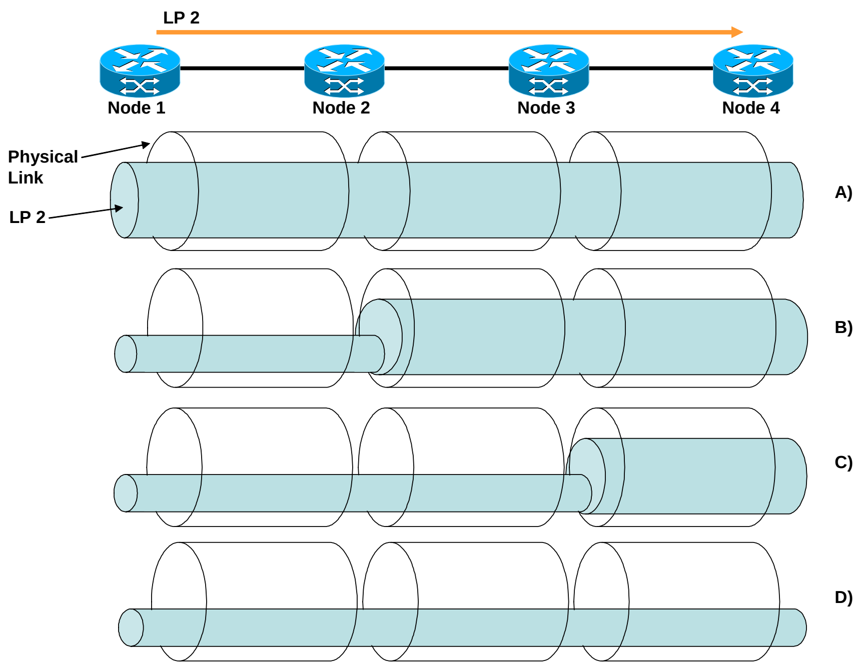


Fig. 4.17: Detail of the Mechanism for Reducing the Bandwidth of an LP.

In this FNO algorithm the selection of an underused LP in order to reduce it and assign its bandwidth to a congested LP can only take place in the origin node. This means that in the transit nodes the algorithm acts in exactly the same way as the FBO algorithm, only considering the free bandwidth in the links.

If there are several candidate LPs which are underused and have enough bandwidth to give to the congested LP, choosing which of them to use could follow several criteria. For instance, it could be selected randomly, or the shortest LP could be chosen first (minimisation of the messages), or depending on its connections history and forecast, etc.

Any-LP (ALP): In this case, in all the nodes the congested LP traverses, it is possible to assign free bandwidth from the links and also unused bandwidth from any other LPs coinciding in one or more LP spans with the congested LP, independent of their initial node. In a given node of the congested LP route, the algorithm first checks the possibility of using free bandwidth from the link; if this is not possible it checks the possibility of using underused bandwidth assigned to LPs beginning at that particular node; if this second option is also not possible, the algorithm checks other LPs not beginning in that node. The procedure consists in asking the P agent in the initial node of the selected LP, for a reduction in its assigned bandwidth.

Figure 4.18 illustrates the situation. Supposing LP1 is congested, Pa2 will initially check the free bandwidth in link 2-3. If there is not enough free bandwidth then it considers the reduction of an LP that begins at node 2. The candidates are LP2 and LP3 (situation A). If it is not possible to reduce any of these LPs then Pa2 considers other LPs that go through the physical link 2-3 and do not begin in node 2, e.g. LP4 and LP5 that both begin at node 1 (situation B). In this case Pa2 sends an *AskForBW* message to Pa1 asking if it is possible to reduce one of them and use the bandwidth in order to increase LP1. Note that in this case Pa2 probably does not have up-to-date information on the occupation of the LPs beginning at node 1, since this information is kept up-to-date only by Pa1's partial network view. Pa2 waits for the response from Pa1 and if it can reduce one of the two LPs then Pa1 will perform a reduction process (*ReduceBW* message) that will go through node 2. Pa2 detects the *ReduceBW* message and can reserve the bandwidth in link 2-3 and start to increase LP1.

Suppose now that the LP1 increasing process (the *IncreaseBW* message) has just arrived to Pa3. In the ALP algorithm Pa3 must also check all the possibilities in order to increase the last LP1 span. Thus it must check the free bandwidth in link 3-6, if there is not enough then it checks whether there is an LP beginning at node 3 which could be reduced, (LP6 in Figure 4.18 C), and if this is also not possible then it checks any other LPs sharing the physical link 3-6 (LP5 in Figure 4.18 D).

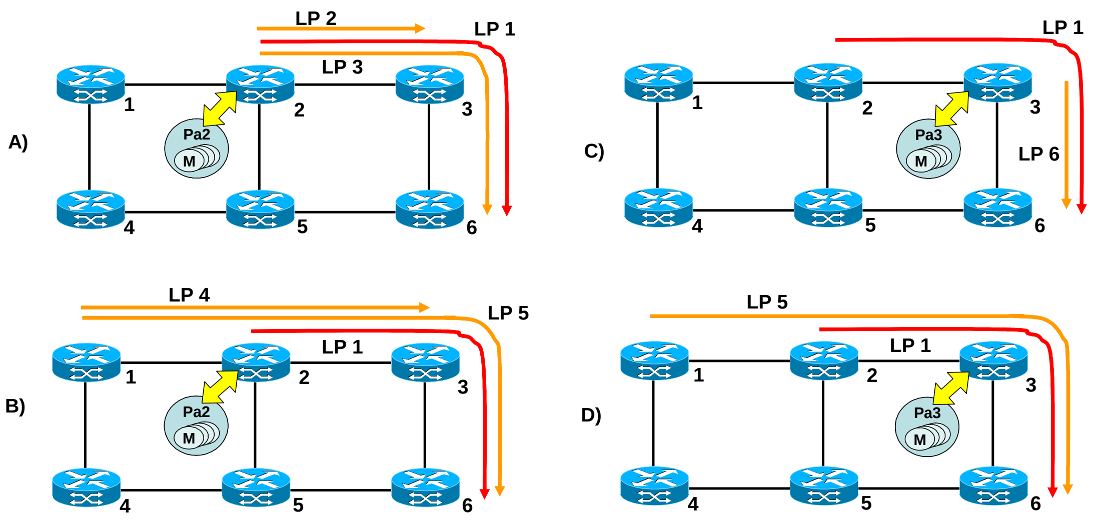


Fig. 4.18: Different Situations and the Corresponding Candidate LPs using the ALP algorithm.

Figure 4.19 shows the messages between the P agents in this example using the ALP algorithm. In this example Pa2 asks Pa1 for bandwidth in link 2-3, Pa1 then starts a reduction of the bandwidth assigned to LP2. This reduction process, the *ReduceBW* messages, also gives information about the reduction motivation and this makes it possible for Pa2, once it has received the reduction message, to start the increasing procedure on the LP1 bandwidth. This is possible because the reduction procedure, once started at the initial node, cannot be rejected by intermediate nodes. Note that the *AskForBW* mechanism also ends with a Response message, although in this case this would not be strictly necessary.

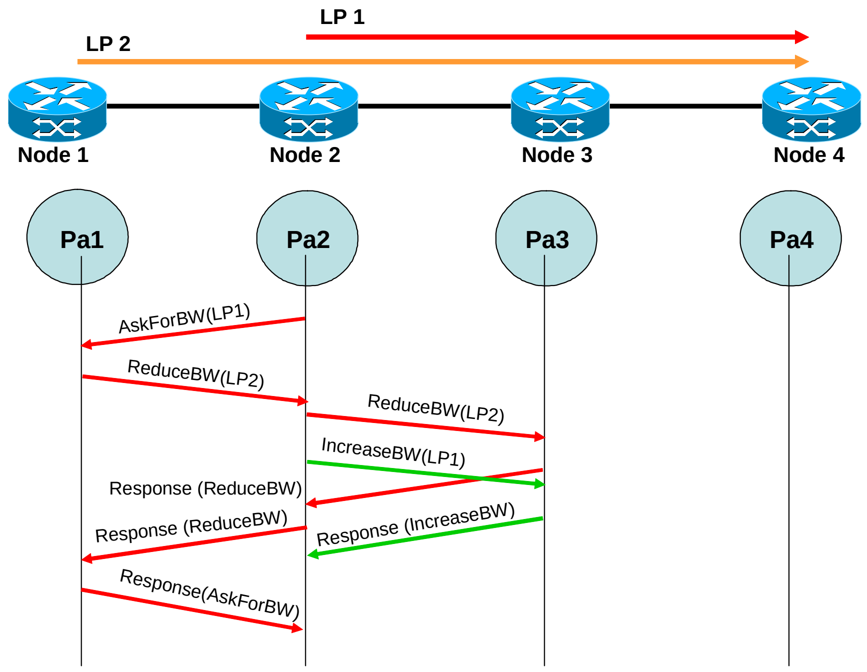


Fig. 4.19: Example of an Interchange of Messages in the Case of the ALP Algorithm.

We have modelled these three distributed algorithms as defined in this section. The idea is that there is an exact copy of the given algorithm in each node. The algorithm could be activated by the detection of a congested LP by the TF mechanism, and also when a message arrives from another P agent from another node. The messages go up and downstream, the congested LP being the reference.

It is necessary to define several terms and properties:

δ is the fixed amount of bandwidth used to increment/decrement an LP,

N_P is the node p,

$\mathcal{N}$ is the set of all the nodes, $N_p \in \mathcal{N} \quad \forall p$

L_j is the link j, defined as a list, $L_j = (s_j, d_j, b_j, u_j)$, where:

s_j is the source node of link L_j,

d_j is the destination node of link L_j,

b_j is the bandwidth of link L_j, and

u_j is the used bandwidth of link L_j, bandwidth already assigned to LPs.

$\mathcal{L}$ is the set of all the links, $L_j \in \mathcal{L} \quad \forall j$

P_i is the Logical Path i, defined as a list $P_i = (R_i, L_{i,1}, L_{i,2}, ..., L_{i,R}, B_i, U_i)$, where

R_i is the number of links that traverse the LP i, and $1 \leq R_i \leq \; |\mathcal{L}|$

$L_{i,k}$ is the k link of the LP i. This is a 'local' numeration with respect the LPs, thus

$L_{i,k} \in \mathcal{L}$, and, $\forall k, \exists! j \,|\, L_{i,k} = L_j$

B_i is the allocated bandwidth of LP i

U_i is the used bandwidth of LP i, bandwidth already assigned to user connections.

$\mathcal{P}$ is the set of all the LPs, $P_i \in \mathcal{P} \quad \forall i$

$\mathcal{P}_j^s$ is the set of all LPs that start in the link L_j and $\mathcal{P}_j^s = \left\{ P_i \,|\, L_{i,1} = L_j \right\}$

$\mathcal{P}_j^t$ is the set of all LPs that go through link L_j and $\mathcal{P}_j^t = \left\{ P_i \,|\, \exists k, L_{i,k} = L_j \right\}$

We have also defined the following functions to access / modify the values of the elements of a Link (L_j) or of an LP (P_i). Note that these functions can be used to obtain the value and also to set the value, depending on whether the function is called after or before the assignation operator ($\leftarrow$) respectively.

$\mathrm{s}(L_j) = s_j$	returns the source node number of link L_j.
$\mathrm{d}(L_j) = d_j$	returns the destination node of link L_j.
$\mathrm{b}(L_j) = b_j$	returns the bandwidth assigned to link L_j.

$u(L_j) = u_j$	returns the used bandwidth of link L_j.
$R(P_i) = R_i$	returns the number of links of LP P_i.
$\mathrm{Link}(P_i, k) = L_{i,k}$	it returns Link k of LP P_i.
$B(P_i) = B_i$	returns the bandwidth assigned to LP P_i.
$U(P_i) = U_i$	returns the used bandwidth of LP P_i.

We have identified some properties of the previous definitions as follows: in successive links of a given LP the destination node of the previous link must coincide with the source node of the next link.

$$\forall L_{i,k}, L_{i,k+1} \in P_i \quad d(L_{i,k}) = s(L_{i,k+1}) \qquad \text{(Formula 4.6)}$$

Any LP in the network cannot have assigned to it more bandwidth than any of the bandwidths of the links it traverses.

$$\forall k, \forall P_i \in \mathcal{P} \quad \mathrm{B}(P_i) \le \mathrm{b}(L_{i,k}) \qquad \text{(Formula 4.7)}$$

The addition of the bandwidths assigned to all the LPs that traverse a given link must not be greater than the bandwidth of the link.

$$\forall L_j \in \mathcal{L} \quad \sum_{P \in \mathcal{P}_j^t} \mathrm{B}(P) \le \mathrm{b}(L_j) \qquad \text{(Formula 4.8)}$$

Given the definitions, functions, and properties presented in this sub-section we modelled the algorithms FBO, FNO, and ALP as functions. P_k represents the congested LP, i is the segment number of the LP P_k, therefore when the TF detects congestion, e.g. in LP P_3 then it calls increaseBW(P_3,1). The function "wait_response" stops the procedure until the corresponding response message arrives.

Figure 4.20 presents the FBO algorithm. Figure 4.21 presents the FNO algorithm and the same Figure, replacing line number 22 with the one in Figure 4.22, also represents the ALP algorithm.

```
increaseBW( $P_k$, i )
  if (b(Link($P_k$,i)) – u(Link($P_k$,i)) $\geq \delta$ )                ;enough free Bw?
    u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) + $\delta$               ;pre-reservation
    if (i<R($P_k$))                                             ;not the last LP span?
      send_message_downstream(increaseBW($P_k$,i+1))
      wait_response($P_k$, response)
      if (response==increaseBW_succeeded)
        B($P_k$) $\leftarrow$ B($P_k$)+ $\delta$                                  ;increase the LP span
        u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$             ;release the pre-reserv.
        if (i>1)                                                ;not the initial node
          send_message_upstream($P_k$,increaseBW_succeeded)
      else
        u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$             ;release the pre-reserv.
        if (i>1)                                                ;not the initial node
          send_message_upstream($P_k$,increaseBW_failed)
    else                                                        ;the last LP span
      B($P_k$) $\leftarrow$ B($P_k$)+ $\delta$                                    ;increase the LP span
      u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$               ;release the pre-reserv.
      if (i>1)                                                  ;not the initial node
        send_message_upstream($P_k$,increaseBW_succeeded)
  else
    if (i>1)                                                    ;not the initial node
      send_message_upstream($P_k$,increaseBW_failed)
```

Fig. 4.20: The FBO Algorithm.

```
1:increaseBW( $P_k$, i )
2:  if (b(Link($P_k$,i)) – u(Link($P_k$,i)) $\geq \delta$ )           ;enough free Bw?
3:    u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) + $\delta$           ;pre-reservation
4:    if (i<R($P_k$))                                          ;not the last LP span?
5:      send_message_downstream(increaseBW($P_k$,i+1))
6:      wait_response($P_k$, response)
7:      if (response==increaseBW_succeeded)
8:        B($P_k$) $\leftarrow$ B($P_k$)+ $\delta$                           ;increase the LP span
9:        u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$         ;release the pre-reservation
10:       if (i>1)                                             ;not the initial node
11:         send_message_upstream($P_k$,increaseBW_succeeded)
12:     else
13:       u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$         ;release the pre-reservation
14:       if (i>1)                                             ;not the initial node
15:         send_message_upstream($P_k$,increaseBW_failed)
16:   else                                                     ;the last LP span
17:     B($P_k$) $\leftarrow$ B($P_k$)+ $\delta$                             ;increase the LP span
18:     u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$           ;release the pre-reservation
19:     if (i>1)                                               ;not the initial node
20:       send_message_upstream($P_k$,increaseBW_succeeded)
21: else                                                       ;find an LP to be reduced
22:   if (i==1 $\wedge$ search($P_q$ | $L_j$=Link($P_k$,1) $\wedge$ $P_q \in \mathcal{P}^s_j$ $\wedge$ B($P_q$)-U($P_q$)$\geq \delta$ ))
23:     send_message_downstream(decreaseBW($P_q$, 1))
24:     wait_response($P_q$, response)                         ;$P_q$ is now decreased and
25:     u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) + $\delta$           ;pre-reservation is possible
26:     if (i<R($P_k$))                                        ;not the last LP span?
27:       send_message_downstream(increaseBW($P_k$,i+1))
28:       wait_response($P_k$, response)
29:       if (response==increaseBW_succeeded)
30:         B($P_k$) $\leftarrow$ B($P_k$)+ $\delta$                         ;increase the LP span
31:         u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$       ;release the pre-reservation
32:         if (i>1)                                           ;not the initial node
33:           send_message_upstream($P_k$,increaseBW_succeeded)
34:       else
35:         u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$       ;release the pre-reservation
36:         if (i>1)                                           ;not the initial node
37:           send_message_upstream($P_k$,increaseBW_failed)
38:     else                                                   ;the last LP span
39:       B($P_k$) $\leftarrow$ B($P_k$)+ $\delta$                           ;increase the LP span
40:       u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) - $\delta$         ;release the pre-reservation
41:       if (i>1)                                             ;not the initial node
42:         send_message_upstream($P_k$,increaseBW_succeeded)
43:   else
44:     if (i>1)                                               ;not the initial node
45:       send_message_upstream($P_k$,increaseBW_failed)


1:decreaseBW( $P_k$, i )
2: B($P_k$) $\leftarrow$ B($P_k$) - $\delta$
3: u(Link($P_k$,i)) $\leftarrow$ u(Link($P_k$,i)) + $\delta$
4: if (i < R($P_k$))
5:   send_message_downstream(decreaseBW($P_k$,i+1))
6:   wait_response($P_k$, response)
7:   if (i>1)send_message_upstream($P_K$, decreaseBW_succeeded)
```

Fig. 4.21: The FNO Algorithm.

```
22:   if (search ($P_q$ | $L_j$=Link($P_k$,i) $\wedge$ $P_q \in \mathcal{P}^t_j$ $\wedge$ B($P_q$)- U($P_q$)$\geq \delta$ )
```

Fig. 4.22: The ALP algorithm (replace this line in the FNO Algorithm).

There is an initial set of messages sent between the P agents. These messages are presented in Table 4.2. There are more messages defined in the next section. Note that all the messages sent between P agents carry the information about why an action is requested, i.e. what the detected problem is, as well as the information on the sender P agent's partial network view attached to it.

Message	*Description*
IncreaseBW	Message sent to request an increase of bandwidth for a congested LP. The action can be completed or not.
DecreaseBW	Message sent to decrease the bandwidth of an LP, which usually would be underused
AskForBW	Message sent to request a bandwidth decrease on an LP that starts on a different node but goes through the node that sends this message.
Response	Response message for all the previously defined messages. It carries the response type and the action result

Table 4.2: Bandwidth Reallocation Messages Between P agents

It is important to note that whichever algorithm is used it is possible that the attempt to increase a congested LP can finish without success. In that case there are two possibilities: the first one is to do nothing and wait, and in this case two things can happen, the LP is no longer congested or if it remains congested the P agent can keep trying to increase it; the second possibility is the rerouting of the congested LP or another one with the objective of making room.

4.4.3 Logical Path Rerouting

This section describes the proposed algorithm for the P agents with the objective of rerouting an LP. The idea of the rerouting mechanism is to use it when the bandwidth reallocation mechanisms fail in the increasing bandwidth process of a congested LP. In this case, it is possible to reroute the congested LP through another physical path with enough free bandwidth to enlarge it, or it is also possible to reroute another LP to make room in the physical path of the congested LP.

We decided to only use the rerouting of the congested LP, to minimise the messages between P agents. This is because when the rerouted LP is the congested one, to find an alternative path that fulfils the requirements of enough free bandwidth means that a single reroute of the congested LP will solve the problem. However, and mostly due to the possibility of the P agents' partial network view being out of date, trying to make room in the congested LP path means that it might be necessary to reroute more than one LP. This will help to achieve a better overall system scalability.

There is a special characteristic of the rerouting mechanism in this situation. That is that it is not necessary to use a fast or real-time routing mechanism because it is not the case of establishing a new connection. On the contrary, the LP already exists and it is dispatching the user connections, there is only the problem that it is congested and the bandwidth reallocation mechanisms cannot be applied. Although it is important to achieve a good rerouting time in order to accept the connections that could be being rejected, this number of rejected connections should not be very high so as not to allow a few seconds for the rerouting operation. Therefore, the offered traffic pattern will not change so abruptly that makes lose a large amount of user connections.

Another interesting characteristic is that when the bandwidth reallocation mechanisms cannot be applied to a given congested LP, this is also a symptom that one or more physical links in that path are full, and this indicates that there is a high probability that other LPs traversing that particular link or links may also have problems trying to increase their bandwidth in the case that they become congested. In other words, rerouting the congested path, apart from enabling its bandwidths to increase, could also help to alleviate a specific physical link and keep the network better balanced.

In order to reroute the congested LP it is possible to use the many routing algorithms that are proposed in the literature. Typically it could be interesting to use a constraint based routing mechanism. For instance the Shortest Widest Path, or the Widest Shortest Path [Guerin et al., 1997], are good candidates. However there is the problem of the incomplete network status view and that P agents only have a partial network view, which may not be up to date. In other words, each P agent maintains a complete Logical Network view but this view usually is out of date. This means for instance that a P agent could know that some time ago there was an LP, with a certain amount of bandwidth assigned to it, going through a specific link of the network, however it does not know if this is currently true at the time of rerouting.

In such a situation we have decided to apply the following procedure. When a P agent decides to apply the rerouting of a congested LP, then it calculates the new route using a given routing algorithm and its partial network view. If this routing procedure gives several possible routes they are ordered according to several criteria and one of them is selected. Then the P agent initiates the rerouting mechanism by pre-reserving in the first link the required bandwidth for the rerouted LP, and sending a *Reroute* message to the next P agent of the selected route. This *Reroute* message includes all the routing information that the initial P agent has calculated, including the selected route, and also attaches the P agent’s partial network view.

The P agent that receives a *Reroute* message acts as follows. First it updates its partial network view as usual, and then it recalculates the rest of the route using its up to date partial network view. It does not matter whether the rest of the recalculated route coincides or not with the received route of the previous P agent, the new route has been calculated with more information and therefore is more up to date, therefore this updated route is the one followed. Moreover, this route is shorter, and the shorter the route, the more precise the partial network view the P agent has. This means that the route is recalculated at each node until the destination node is reached. On the way back the rerouted LP is established.

The final path of the rerouted LP may or may not coincide with the original route calculated by the P agent at the initial LP node. The rerouted LP may also partially coincide with the original route of the congested LP.

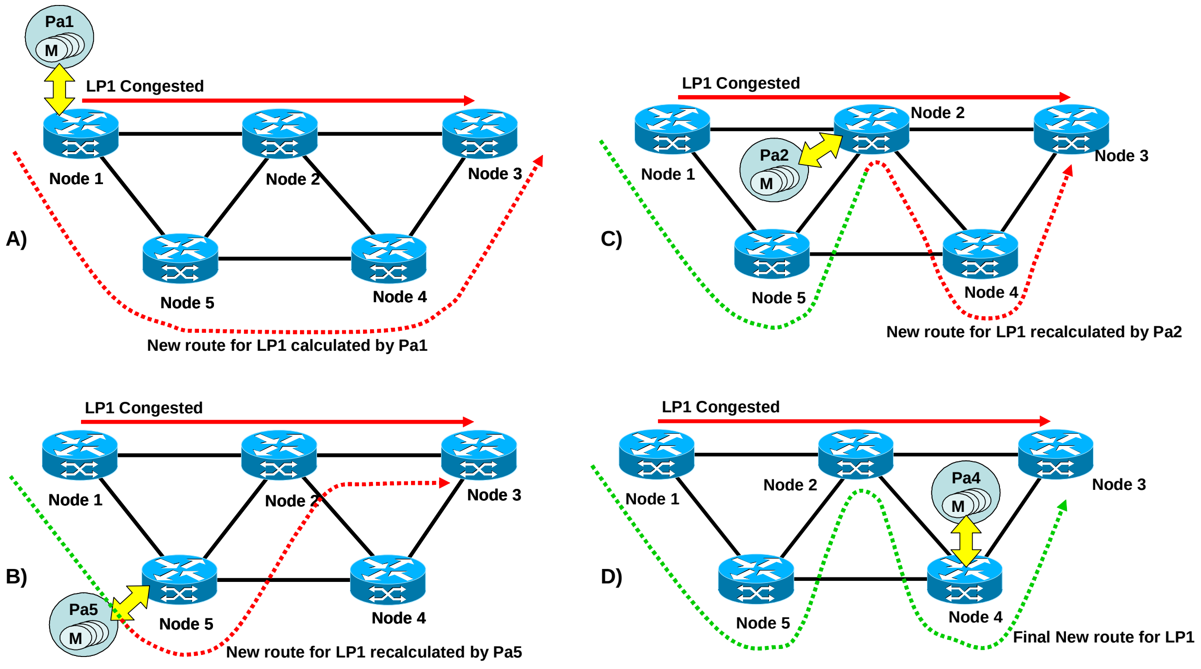


Fig. 4.23: Example of a Rerouting Process. Every P agent in the Route Recalculates it.

Figure 4.23 shows the process of rerouting a congested path for LP1. Using the selected routing algorithm, Pa1 calculates the route 1-5-4-3 and sends a *Reroute* message to Pa5. Pa5 recalculates the route from 5 to 3 and using its updated information changes the overall route to 1-5-2-3, and then sends the *Reroute* message to Pa2. Pa2 knows the real situation of link 2-3 and it is not possible to establish the new route through this link. In any case Pa2 must recalculate the route and therefore it changes again to 1-5-2-4-3. Supposing there are no problems in link 4-3, Pa4 establishes the last LP span of the new route for LP1 and sends a positive *Response* message upstream, which is propagated to the initial node and the new LP1 is established. It is possible that in some links the new rerouted LP1 may coincide with the old LP1. In that case the bandwidth needed by the new LP1 includes the bandwidth of the old LP1.

Once the new route is established the ongoing user connections of the old route must be switched to the new route. This is done in node 1. When the original LP1 becomes empty, and in the case that the original LP1 does not coincide with the rerouted LP1, Pa1 sends a *Release* message downstream and the original LP1 is released (otherwise it is released only in some links), becoming all the bandwidth that was assigned to it available. Therefore there are two new messages to add to the table of messages between P agents: *Reroute* and *Release*.

It could also happen that the reroute process reaches a node belonging to the new route where there is no possibility of continuing on to the destination node, i.e. the routing algorithm returns void. In such a case, there are two possibilities.

- The first one is returning all the way back to the initial node with negative *Response* messages and aborting the rerouting. In this case nothing, neither bandwidth reallocation nor rerouting, can be done in order to increase the congested LP bandwidth. The only possibility would be waiting and to keep on trying. However, it is not interesting to try again and again continuously and it would be better to wait some time ignoring the possible alarms from the M agent in charge of the congested LP. This is because if it is not possible to do anything for a congested LP and the P agent keeps trying then there will be a lot of message in the network for nothing. The attempts should be conditioned to waiting time intervals and the detection again of congestion in the LP.
- The second possibility is that the routing algorithm calculates several possible routes in every node. These routes can then be classified and one selected to follow at first. When the *Reroute* message arrives to a node where there is no possible route to the destination, then a negative *Response* message is sent back until it finds a node with an alternative route. However, before proceeding with the old alternative route P agents act as usual first updating their partial network view and then recalculating the possible routes. Figure 4.24 shows an example. The initial situation A, not depicted, is the same as in Figure 4.23. When the *Reroute* message arrives to Pa2 it finds in this case that there is no possible route to node 3 coming from node 5. Thus a negative *Response* message is sent back to Pa5, which recalculates the possible routes using the information it already has plus the information that has just arrived from node 2. In the example, Pa5 finds a second possibility and resumes the rerouting

process through node 4. It could be interesting to limit the alternatives to some number in order to limit the search in width and also in time.

An important issue is that while the rerouting process is in progress the status of the congested LP being rerouted becomes special in the sense that the possible congestion alarms are ignored and no action can be carried out through that LP.

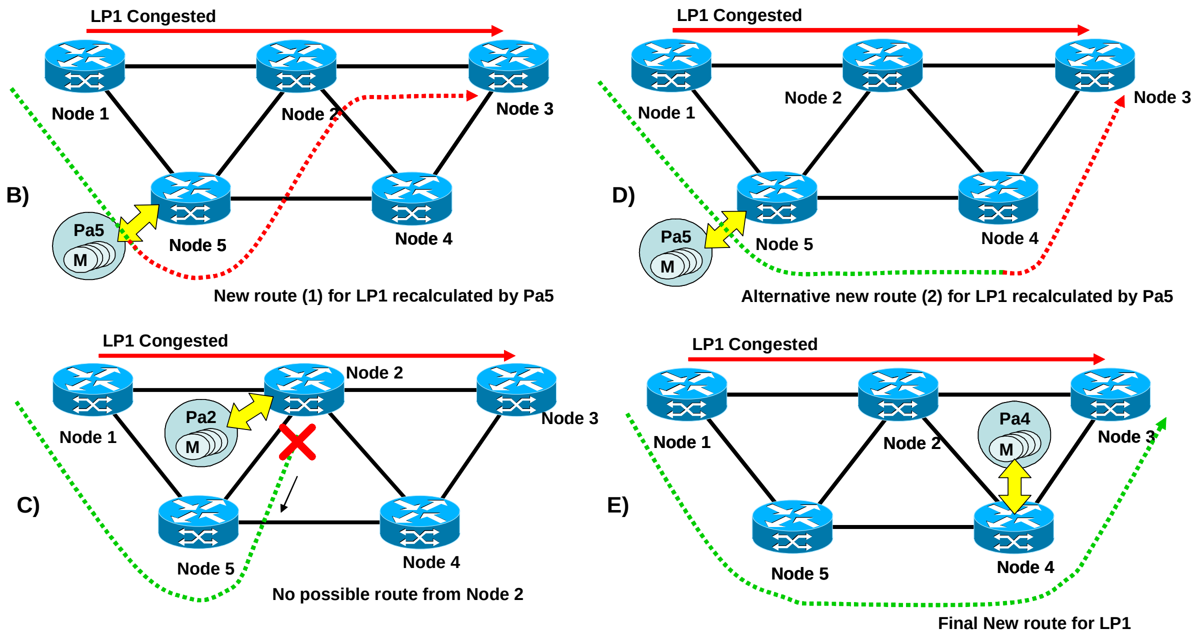


Fig. 4.24: Example of Rerouting with Alternative Routes.
Situation A) is the Same as in Figure 4.23.

The algorithm for the rerouting process is presented in Figure 4.25 in a similar form as the bandwidth reallocation algorithms. The function Reroute is executed when a P agent decides to reroute an LP or when it receives a *Reroute* message. The parameters are the hop number, the destination node, the LP to reroute, and the desired capacity. Note the function “wait_response” waits until the positive or negative *Response* message arrives. The first P agent executes Reroute(1, Destination_node, LP rerouted, desired capacity).

As stated before any routing algorithm could be used. It is not the objective of this thesis to investigate routing algorithms, however we propose a routing algorithm that balances the search of a short route fulfilling the constraint of having enough free bandwidth for the rerouted LP taking into account the fact that the farther the link is from the current node the higher the possibility that the information of the current partial network view is out of date.

```
Reroute( i, N_d, LP, C)
  j ← 0
  continue ← false
  repeat
    find a list of possible routing alternatives using the routing algorithm and the
                                                           actualised partial network view.
    if (routes found)
      order the route list with some criteria and select the first one
      send_message_downstream(Reroute(i+1,N_d , LP, C))
      wait_response
      if (positive response)
        establish the current LP span
        continue ← true
        if (i>1)
          send_message_upstream(N_d, Reroute_succeeded)
      else
        j ← j+1
    else
      continue ← true
      send_message_upstream(N_d, Reroute_failed)
  until ((j == Alternative_Routes_Limit) or (continue == true))
```

Fig. 4.25: Rerouting Algorithm.

The routing algorithm has two phases. The first one builds an adjacency matrix with negative values (-1) for the nodes that do not have physical links between them, and with the amounts of free bandwidth it is supposed that each link has deduced from the P agent's partial network view. The second step uses this adjacency matrix to calculate all the possible routes to the destination node and order them. This means to only consider the routes formed by links where there is enough free bandwidth for the rerouted LP. The order of the list of possible routes is based on a weight φ_R that is:

$$\varphi_R = \omega_1 \cdot \frac{1}{H} + \omega_2 \cdot \frac{\sum_{i=0}^{H-1} (H-i) \cdot p_i}{\max(p_i) \cdot \sum_{i=0}^{H-1} (H-i)} \qquad \text{(Formula 4.9)}$$

where:

- ω_1 and ω_2 are two defined values to divide the importance of the first and second terms of the addition. The first term gives a measure of the route length and the second is a weighted mean of the available free bandwidth.
- R is the route in which weight φ_R has been calculated
- p_i is the free bandwidth of link i of the route R. The bandwidth assigned to the LP being rerouted also counts as free bandwidth.
- H is the number of hops of route R.

The following example shows how this value is calculated. Consider the situation in Figure 4.26. Suppose all the physical links have a capacity of 10 bandwidth units. There are 5 established LPs of bandwidths 2, 3, 4, 5, and 7 respectively, LP1 (initial bandwidth of 3 units) is congested and Pa1 decides to try to increase it to 5 units. Supposing a bandwidth reallocation is not possible then Pa1 decides to reroute LP1. First of all, it constructs an adjacency matrix like the one in Table 4.3. Each cell contains the free bandwidth of the corresponding link (bandwidth not assigned to LPs, except LP1, and not reserved for restoration purposes). The data in the table comes from the partial network view of Pa1 and may be out of date. We have supposed in this example that Pa1 has the correct information.

Using this adjacency matrix all the possible routes between node 1 and node 3 which can allocate 5 units to the rerouted LP1 are calculated. This list is displayed in Table 4.4. Then the value φ_R is calculated (also in Table 4.4) for each route, the route with the greatest value φ_R is the one selected. In this case, with $\omega_1 = \omega_2 = 1$, the selected route is R_a. Note that if R_a was not possible then the selected route was R_b, which is longer but it has more free bandwidth than R_c.

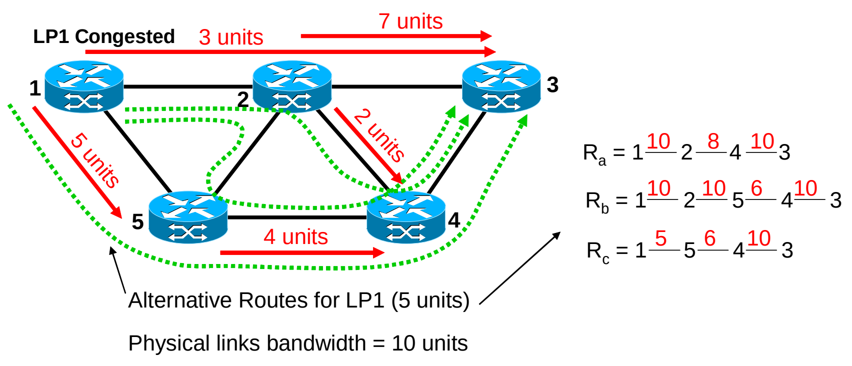

LP1 Congested
3 units
7 units
1
2
3
4
5
5 units
2 units
4 units
$R_a = 1 \overset{10}{—} 2 \overset{8}{—} 4 \overset{10}{—} 3$
$R_b = 1 \overset{10}{—} 2 \overset{10}{—} 5 \overset{6}{—} 4 \overset{10}{—} 3$
$R_c = 1 \overset{5}{—} 5 \overset{6}{—} 4 \overset{10}{—} 3$
Alternative Routes for LP1 (5 units)
Physical links bandwidth = 10 units

Fig. 4.26: Routing Example of LP1 from Node 1 to Node 3.

		Destination Node				
		1	**2**	**3**	**4**	**5**
Origin Node	**1**	-	10	-	-	5
	2	10	-	3	8	10
	3	-	10	-	10	-
	4	-	10	10	-	10
	5	10	10	-	6	-

Table 4.3: Example of an Adjacency Matrix Calculated by Pa1 for rerouting LP1

Routes	*Hops (H)*	*max(p_i)*	*φ*
Ra = 1-2-4-3	3	10	1.26
Rb = 1-2-5-4-3	4	10	1.17
Rc = 1-5-4-3	3	10	0.95

Table 4.4: Results of the Routing Example. $\omega_1 = \omega_2 = 1$

4.4.4 Fast Restoration Mechanism and Spare Capacity Assignment

This sub-section describes the proposed mechanisms for the pre-planned restoration. As it is described in chapter 3, we focus only on pre-planned restoration based on backup LPs, and more specifically on end to end disjoint backups. We also, at the same time, set the protection level to a single link failure which allows reserved bandwidth to be shared between several backup LPs, with the objective of saving bandwidth.

It is not the goal of this thesis to research the problem of logical network planning, i.e. the establishment of the initial set of LPs and backup LPs. There are many works on this area of constraint based routing of both the working LP and the backup LP. Like the bandwidth reallocation mechanisms, the starting point is a network that is already established, composed of the working LPs and the backup LPs. This section refers only to the mechanisms implemented by the P agents in order to perform the switchover when a fault occurs. It also focuses on the shared bandwidth requirements. The coordination of the mechanisms described in this section with the bandwidth reallocation and LP rerouting mechanisms is described in section 4.5.

During normal operation the backup LPs are not used. As Figure 4.6 shows, when a fault occurs impacting a physical link, the Node Control System sends an alarm indication to the P agent residing in the same node. If the situation is that the P agent resides at the same node or at another computer, and the communications with the Node Control System are performed through an SNMP agent, then an SNMP Trap message should be programmed in case of failure and captured by the P agent.

On receiving the failure alarm, the P agent sends a *LinkFailed* message to all the nodes that are the origin of at least one LP traversing the failed physical link. This message is not replied to using *Response* messages. Using this mechanism, every P agent that controls a node, which is the origin of one or more of the LPs that traverse the failed link, receives a *LinkFailed* message. When a P agent receives this message it sends an alert to every M agent responsible for an LP that traverses the failed link. The M agent sends a *CaptureBW* message, which is also not answered, along the backup LP in order to capture the already pre-reserved bandwidth. Figure 4.27 shows an example of these actions. If the pre-reserved bandwidth for the backup LPs is not being used for other low-priority traffic the switchover action and the sending of the *CaptureBW* message downstream along the backup path can be almost simultaneous because the bandwidth is already captured and this message is only to inform the nodes that the backup is going to be used.

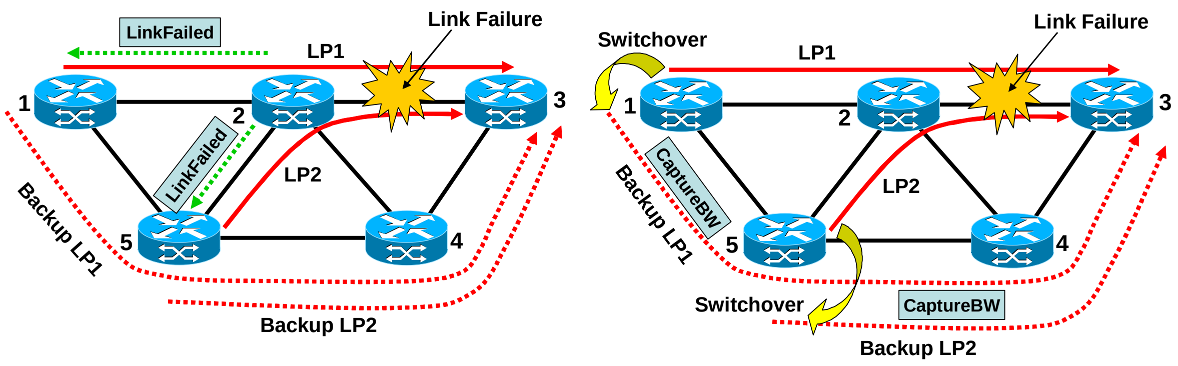


Fig. 4.27: Failed Link Notification and Bandwidth Capture for the Backup LPs.

While a physical link is inactive due to failure and the backup LPs are being used, the original working LPs are not released. Moreover, a similar mechanism is also proposed in order to restore the previous situation when the physical link becomes active again. It is supposed that the P agent responsible for the failed link receives an indication when the link is restored. After that, the P agent sends *LinkRepaired* messages to the origin nodes of the LPs that traverse the failed link. On receiving these messages the user connections are switched back to the original LP and the captured bandwidth of the backup LPs is returned to the previous situation: spare bandwidth reserved for backups in case of failures. Figure 4.28 shows this procedure.

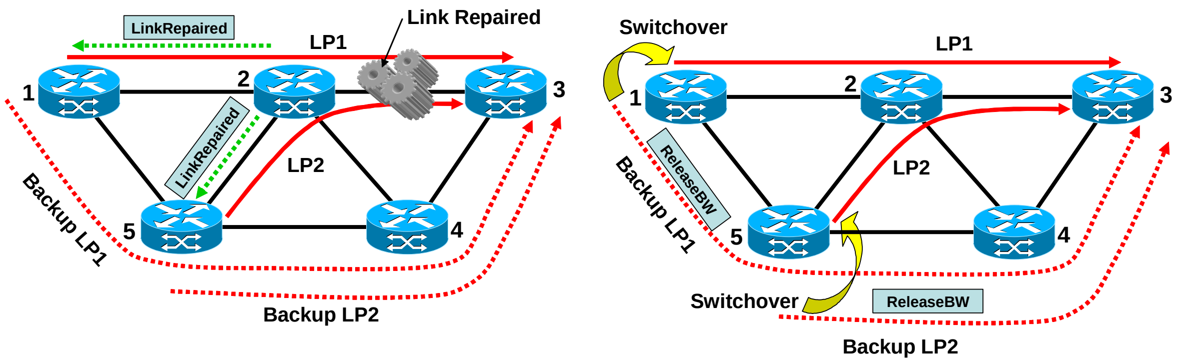


Fig. 4.28: Repaired Link Notification and Bandwidth Release of the Backup LPs.

P agents are responsible for maintaining the status of the outgoing physical links of the nodes where they are placed. Every outgoing link has an available capacity which is usually divided into three blocks (Figure 4.29): Free bandwidth not assigned to any purpose, Reserved bandwidth assigned to the working LPs, and Spare bandwidth reserved for the backup LPs in case of failures.

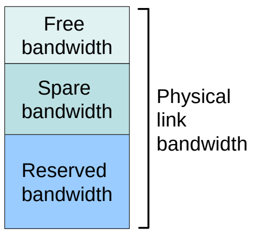


Fig. 4.29: Physical Link Bandwidth Partition.

The Spare bandwidth can be shared for several backups provided that the desired protection level is established, for instance, against one simultaneous link failure. Depending on the desired protection level, and the distribution of the working LPs and backup LPs, a certain amount of spare bandwidth is required in the links, which can be shared to a certain degree. For instance, if the desired protection level is set against one simultaneous link failure then, it is possible to calculate the required Spare shared bandwidth on a link from the following formula:

$$B(L_{x-y}) = \max\left\{b[L_{x-y}, L_{i-j}]\right\} \qquad \forall L_{i-j} \text{ physical link} \qquad \text{(Formula 4.10)}$$

where $L_{p\text{-}q}$ is a physical link going from node p to node q (do not confuse with the notation used in the bandwidth reallocation algorithms). $b[L_{p\text{-}q}, L_{r\text{-}s}]$ is a matrix where each position is calculated from:

$$b[L_{p-q}, L_{r-s}] = \sum_{\Lambda(L_{p-q})\cap\Gamma(L_{r-s})} \beta_k \qquad \text{(Formula 4.11)}$$

where β_k is the required bandwidth of the LPk, and Λ, Γ are two sets defined as follows:

$\Lambda(L_{x\text{-}y})$ = {LPk | LPk passes through $L_{x\text{-}y}$}

$\Gamma(L_{x\text{-}y})$ = {LPk | the backup of LPk passes through $L_{x\text{-}y}$}

Let us imagine the example situation in Figure 4.30. The required Spare bandwidth in link 1-2 depends on the backup LPs that go through this link and the possible failure situations. There are three backup LPs (bLP): bLP1, bLP2 and bLP5. Then, we have to see what happens when link 1-5 fails. This failure affects LP1 and LP2, therefore both their backups must be activated requiring their bandwidth. Thus these backups cannot share the bandwidth. Supposing now that the failed link is 5-4, this affects LP2, LP3, and LP5. In this situation the required bandwidth in link 1-2 is the addition of the bLP2 and bLP5, which also cannot share the bandwidth. Table 4.5 shows this numerical example supposing the bandwidths of LP1, LP2, LP3, LP4, and LP5 are respectively 5, 3, 2, 7, and 4 units.

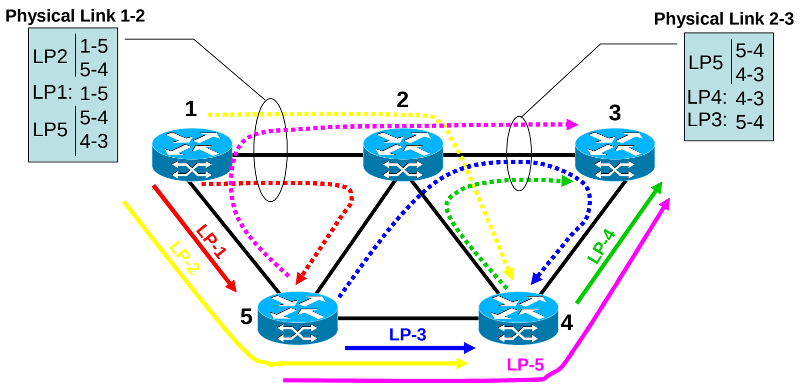


Fig. 4.30: Example Situation for the Required Spare Bandwidth in the Links.

Physical Link	*Protects*	*Capacity*
1-2	LP1 (1-5)	5 units
	LP2 (1-5-4)	3 units
	LP5 (5-4-3)	4 units

Failed Link	*Active Backups*	*Required Bw*
1-5	LP1 & LP2	5+3 = 8 units
5-4	LP2 & LP5	4+3 = 7 units
4-3	LP5	4 units
	Required Shared Bw:	8 units

Table 4.5: Required Bandwidth in Link 1-2 in several Failure Situations.

Using the formulas previously presented with this same example, the matrix $b[L_{p\text{-}q}, L_{r\text{-}s}]$ is presented in Table 4.6 (all the rows are presented but only the columns for the links where the LPs are placed are shown). For instance $\Lambda(L_{1\text{-}5})$ = {LP1, LP2} and $\Gamma(L_{1\text{-}2})$ = {LP1, LP2, LP5}. Therefore $\Lambda(L_{1\text{-}5}) \cap \Gamma(L_{1\text{-}2})$ = {LP1, LP2} and the value of the matrix corresponding to the position $b[L_{1\text{-}5}, L_{1\text{-}2}] = 5 + 3$, the bandwidths of LP1 and LP2. Another example using $\Lambda(L_{5\text{-}4})$ = {LP2, LP3, LP5} and the previously defined $\Gamma(L_{1\text{-}2})$, is $\Lambda(L_{5\text{-}4}) \cap \Gamma(L_{1\text{-}2})$ = {LP2, LP5} and the value of the matrix corresponding to the position $b[L_{5\text{-}4}, L_{1\text{-}2}] = 3 + 4$. In the table the first index gives the column and the second the row. The shared Spare bandwidth ($B(L_{x\text{-}y})$) required in each link is given by the maximum of each row. The columns of the matrix with only zeros are omitted.

It is not necessary that every P agent is aware of the required Spare bandwidths in the whole network. A P agent must just know the physical path of all the LPs whose backups go through every outgoing physical link it is responsible for, and the required bandwidth for each backup LP, which is usually set in the backup establishment process.

		L_{4-3}	L_{5-4}	L_{1-5}		$B(L_{x-y})$
L_{1-2}		0	3+4	5+3		8
L_{2-1}		0	0	0		0
L_{2-3}		7+4	2+4	0		11
L_{3-2}		0	0	0		0
L_{3-4}		4	2	0		4
L_{4-3}		0	0	0		0
L_{4-5}	···	0	0	0	···	0
L_{5-4}		0	0	0		0
L_{1-5}		0	0	0		0
L_{5-1}		0	4	0		4
L_{2-4}		0	3	3		3
L_{4-2}		7	0	0		7
L_{2-5}		0	0	5		5
L_{5-2}		4	2	0		4

Table 4.6: Required Spare Bandwidth in the Example.

4.5 Putting Everything Together

In the previous section we detailed the mechanisms that the Multi-Agent System must implement. However we have described them without taking into consideration the possible inter-relationship between them. This is a problem detected in the dynamic resource management environment which is usually not handled. Centralised systems perform periodic (e.g. every hour) readjustments recalculating the whole Logical Network including the working and backup LPs, this problem is not presented in this situation. However, in a distributed dynamic scenario where there can be readjustments automatically, without knowing where and when they will be produced, coordination is necessary between the different mechanisms which have a part in changing the Logical Network.

Due to the fact that the P agents have a limited partial network view, possible problems could arise in an uncoordinated scenario. Some situations that require coordination between the different mechanisms have been identified:

- If the bandwidth of a working LP is increased or decreased the spare bandwidth reserved in several links along the corresponding backup LP must also be increased or decreased accordingly.
- When an LP is rerouted, this affects the required amount of spare bandwidth in several links of the backup path due to Formula 4.10. Also the rerouted working LP and its backup LP must be link-disjoint when using an end-to-end pre-planned restoration mechanism.
- When a failure occurs the user connections are switched over to backup paths. In this situation the backup paths act as normal LPs and they can be increased or decreased due to the bandwidth reallocation and/or be rerouted.

In these situations some rules, constraints and/or mechanisms have to be established in order to avoid interferences between the bandwidth reallocation and rerouting with the pre-planned restoration and the Spare bandwidth management.

As an initial decision, in all the cases where it is possible, we have decided not to introduce new messages to distinguish the working LPs from the backup LPs. Thus the only difference will be an attribute or special stamp on the messages in order to distinguish when the messages refer to a working or a backup LP. All the messages that the P agents send between them are listed in Table 4.7 (some of them have already been listed in previous tables).

Message	*Description*
IncreaseBW	Message sent to request an increase of bandwidth for a congested LP. The action can be completed or not.
DecreaseBW	Message sent to decrease the bandwidth of an LP, which is usually underused
AskForBW	Message sent to request a bandwidth decrease on an LP that starts in a different node but goes through the node that sends this message.
Response	Response message for all the previously defined messages. It carries the response type and the action result
Reroute	Message sent in order to find a new route for a congested LP with the goal of increasing it
Release	Message sent along an old existing LP in order to release it after it has been rerouted
LinkFailed	Message sent upstream to all LPs that go through a physical link that has just failed
LinkRepaired	Message sent upstream to all LPs that go through a failed physical link that has just been repaired
CaptureBW	Message sent downstream of a backup LP indicating it is going to be used instead of the working LP, which has failed
ReleaseBW	Message sent downstream of a backup LP indicating the normal situation is restored and working

Table 4.7: Messages Between P agents

In the case of bandwidth reallocation, it is a different situation when an increase occurs to when a decrease occurs.

- When a decrease occurs the P agent makes the decision at the origin of the LP and, once the decision is made, it is always possible to decrease the bandwidth all along the LP route. The same rule can be applied along the backup LP route. However it has to be taken into account that the spare capacity cannot be decreased straight away because it could be shared with other backup LPs.
- In the case of an LP increase, the backup LP has to be increased accordingly, in this situation there are two possibilities:
    - o The first one is to start the increase of the working and the backup LP simultaneously. It may happen that both can be increased or both cannot be

    increased (not being a consistency problem). When the working path can be increased but the backup path cannot or vice versa the solution is to abort the increasing process and send a message that they cannot be effectively increased (the one that has been successfully increased should be decreased again to its previous bandwidth). This is the option selected for the proposed architecture.
    - The second is to first increase the working LP and if it can be increased then start the backup LP increasing process. This case can only be used if and only if a temporary situation where the working LP is not fully protected can be accepted.

Regarding the rerouting process, there are also several possibilities:

- To perform a rerouting of both the working and the backup LP simultaneously, looking for disjoint routes. However this could be very difficult in a distributed scenario with partial network views. In this case both rerouting processes are also used to increase the bandwidth of the working and backup LPs.
- To fix the backup LP and reroute the working LP or vice versa. That is, reroute only one of them and leave the other fixed. The links that traverse the fixed one cannot be used by the rerouted one and hence market conveniently. In this case the best option is first to increase the bandwidth of the fixed one using a bandwidth reallocation mechanism and second to start the rerouting of the other one.

The option selected for the proposed architecture is the second one, however we made another restriction on the implementation that consists in always leaving the backup LP fixed and only rerouting the working LPs when necessary.

In an exceptional situation of a link failure, when the user connections are using the corresponding backups, we have decided that all the affected LPs and backup LPs cannot be increased nor rerouted. It is expected that the network will be in a delicate situation and it is not convenient to make bandwidth reallocations and rerouting attempts neither in the LPs and backup LPs that traverse the failed link nor in their corresponding backup LPs and LPs. This decision will also save messages between P agents that must put all their attention into the failure situation.

Therefore it is clear that the coordination between the different mechanisms is necessary in such a dynamic and distributed scenario. The proposed rules / heuristics / mechanisms for such coordination represent a first step. The interactions and possible interferences between the different management and control mechanisms can be further investigated. We have only focussed on the resource management mechanisms using the logical network concept in a connection oriented scenario.

Chapter 5 Analysis and Simulation Results

5.1 Overview

This chapter shows the experiments that have been carried out in order to evaluate the proposed architecture's operation. They are focussed on testing how the different proposed mechanisms work and making comparisons between them. We evaluate the capacity of the proposed architecture to perform the joint bandwidth management and fault restoration functions in a distributed way while achieving good performance and scalability. First the different mechanisms were tested separately and after that they were tested together. A model for evaluating the scalability of the proposed architecture is also proposed.

5.2 Introduction

Chapter 4 introduces the proposed architecture based on MAS. It also details the mechanisms and algorithms proposed for every task the architecture must perform. Firstly, in the following sub-section, several tests have been designed in order to compare the different mechanisms proposed for detecting congestion in the LPs and get feedback on their behaviour. Therefore, several experiments have been performed in order to evaluate the different mechanisms individually. After this initial evaluation, the proposed architecture as a whole system has also been tested under several circumstances.

Note that the proposed architecture, for example a congestion detection mechanism, may use just one of the proposed techniques, a combination of them or other mechanisms based on Case Based Reasoning, Fuzzy Logic, Statistical, etc. The main goal of the experiments is to evaluate the behaviour of the proposed architecture in the coordination of the different resource management mechanisms and not the evaluation of every single mechanism. Therefore, the method was to first make some tests in order to validate that every mechanism was operating correctly and obtain some feedback on their behaviour.

All the experiments have been performed using the simulator proposed in [Marzo et al. 2003a]. This is a distributed simulator where every process simulates a network node and these processes can be executed in different computers. Therefore it is possible to implement the proposed MAS that manages the different simulated nodes as well as a distributed system using the proposed architecture. The objective is that the MAS implementation in the simulated network should not be much different than in a real network. As a consequence, the proposed MAS has also been implemented as a real distributed system based on the Java language [Java URL]. Every P agent is an independent Java process, which can have many M agents as threads (lightweight processes). The communication between a P agent and the M agents placed on the same node is performed directly by memory sharing. The communications between different P agents are based on the Java Remote Method Invocation (RMI) and an input message queue with priority is implemented in the P agents. All the messages arriving from the neighbours are queued in an arrival order, except the alarm messages, which have a higher priority. More details about the simulator can be found in Appendix A.

The simulations are performed at a connection level, i.e. the cells or packets are not simulated. The traffic generators simply request the simulated nodes for connections and an admission control is applied, whether the requested connection is accepted or rejected. This admission control mechanism is based on an equivalent bandwidth scheme and 5 different classes of connections (Table 5.1) have been defined. It is possible to use one or more of these traffic classes for the generation of offered connections for each LP. It is also possible to assign different mean interarrival times (negative exponential distributed) and mean duration times (negative exponential distributed) per traffic class and per LP.

Traffic Class	*Equivalent Bandwidth*
1	64 Kbps
2	2 Mbps
3	2 Mbps
4	4 Mbps
5	10 Mbps

Table 5.1: Traffic Classes

All the tests that are presented in the following two sections basically evaluate two parameters: the Connection Blocking Probability (CBP) and the number of messages needed to be sent by the P agents to each other. The CBP is defined as the total amount of rejected connections (R_i) divided by the total amount of offered connections (O_i) for a given LPi:

$$CBP_{LPi} = \frac{R_i}{O_i} \qquad \text{(Formula 5.1)}$$

The average of all the CBP of all the established LPs in the network can be seen as the total CBP of the network. R_i and O_i are absolute values that count from the beginning of the simulation, thus the simulation duration is important when comparing different mechanisms.

In an ideal situation where the logical network is perfectly designed and the offered traffic follows the forecasted parameters exactly, there would be no need for logical network readjustments. Therefore, the proposed MAS would not have to act performing bandwidth reallocations nor rerouting LPs. In that situation there would not be any messages between P agents and only the M agents would be monitoring the conditions of the LPs. However, in real life there can always be unforeseen situations that require readjustments of the logical network as well as network failures. These situations are exceptional and usually a given network can go for periods of, for instance, hours without the need of readjustment. On the other hand, in order to test the proposed MAS and find out how it behaves, we consider it interesting to also test it under unrealistic situations that provoke a large amount of network reconfigurations. That is, we have used logical networks in the experiments that are not well designed for the offered traffic and many changes are required to adapt the logical network. In other cases more realistic scenarios with networks that have peaks of offered load are also used, and these scenarios provoke fewer readjustments and lower connection rejection ratios.

On the other hand some of the experiments are designed in order to confirm that the different algorithms and mechanisms work properly and their individual behaviour is as expected. In these experiments small networks are usually used in order to show the mechanism's operations better. In other cases larger networks are used, for instance when the overall system is tested.

5.3 Separate Evaluation

In this section the distributed mechanisms implemented by our proposed architecture based on MAS are evaluated. In order to better understand their behaviour they have been tested separately. Along with the results we also give their interpretation, possible limitations and improvements on the mechanisms and, in some cases, implementation alternatives. Some of the mechanisms offer better possibilities in their testing than others.

5.3.1 Monitoring and Congestion Detection

This sub-section presents the different experiments that have been performed in order to verify that the TFs proposed as mechanisms for detecting congested LPs are operating properly. The three proposed TFs are *Rejected(t, limit)* (R in the figures), *CBP_{30}(t, limit)* (CBP in the figures), and *Load(t, limit)* (L in the figures). Apart from the use of one of these three TFs and the corresponding limits or thresholds, there are also two other important parameters to be taken into account: the monitoring period and the step size, that is, the time between two measures and the corresponding evaluation of the TF, and the size with which the capacity of the LP is increased when the TF considers it to be congested. Therefore the purpose of these experiments is also to obtain information on the behaviour of the three TFs taking into account the monitoring period and the step size.

Note that lightweight monitoring is desired, not real-time monitoring. Therefore, we propose fast but not real time monitoring using time scales of seconds or tens of seconds. The TFs' design is oriented to this goal, because they are very simple (lightweight) and fit well in the proposed fast but not real time monitoring.

Due to the complete independence of the M agents, it is even possible to monitor different LPs using different TFs in the same network. For this reason we chose to simulate only a two-node network with only one LP. This also helps to focus our attention on the TFs' behaviour, as well as simplifying the simulations and facilitating the possibility of performing many more tests.

The design of the experiment is as follows: A single TF is tested and therefore a simple network with two nodes and only one LP between these two nodes is used (Figure 5.1). In scenario 1 the rerouting is impossible and when congestion is detected in the LP by the TF then the LP capacity is increased by a given amount of bandwidth: the step size. We have chosen to make this increasing operation always possible, defining a very large physical link. In order to evaluate the TFs, it is necessary to provoke congestion in a given LP in order to cause connection rejections. For this reason the simulation starts with a reduced initial LP capacity, which is not capable of accepting the offered load, which is quite high. Therefore, we are interested in the warm up part of the simulation, when there are rejections and the LP is adapting its capacity to the offered load. The performed simulations lasted ten minutes and the effect was that during this time there were many LP capacity changes.

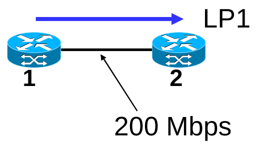


Fig. 5.1: Scenario 1 – TF Tests.

The three TFs have been tested under several limit values, several monitoring periods and several step sizes. All these variables are specified in Table 5.2. The various combinations of these parameters make a total number of 192 different simulations. Moreover, these simulations have been performed with two traffic models: homogeneous and heterogeneous connections. The connection distributions used in both cases are presented in Table 5.3. From the offered connections it has been deduced that at the stable situation the mean offered load is 64 Mbps in the homogeneous connections case and 157 Mbps in the heterogeneous connections case. However, as our interest is in the increasing load period, not in the stable one, and the simulations are 10 minutes long this stable situation is not reached, i.e. the offered load is increasing during these 10 minutes.

Parameter	*Values*			
Monitoring Period	2 s	5 s	10 s	20 s
Step Size	500 Kbps	1000 Kbps	2000 Kbps	4000 Kbps
Rejected Limit	1 connection	3 connections	5 connections	7 connections
CBP Limit	10 %	30 %	50 %	70 %
Load Limit	85 %	90 %	95 %	99 %

Table 5.2: Simulation Parameters

Offered Connections	*Initial LP Capacity*	*Traffic Class (Table 5.1)*	*Mean Inter-arrival Time (negative-exponential distributed)*	*Mean Duration Time (negative-exponential distributed)*
Homogeneous	512 Kbps	1	1 s	1000 s
Heterogeneous	8000 Kbps	1 2	2 s 16 s	1000 s 1000 s

Table 5.3: Offered Connections

The results obtained can be analysed and compared from several points of view. However, because of the different traffic characteristics and offered load the homogeneous and heterogeneous traffic results can only be indirectly compared. Even focussing only on just one traffic type means there are still too many results to be presented in an easily-understood manner. For this reason, the results are grouped in different ways and several graphs are presented showing the behaviour of the TFs with regard to the monitoring interval, the step size and the TF limit. The results displayed in the following figures represent the connection blocking ratio after the ten minute simulation.

TFs behaviour in relation to the Monitoring Period

Firstly, the behaviour of the different TFs and limits regarding the selection of the four different monitoring periods (Table 5.2) are compared. Therefore, once the TF and its limit are selected, it is possible to see that there are no large differences between selecting one or the other monitoring period (Figure 5.2). However, there are some interesting issues to comment on. The shorter monitoring period (2 s) is not always the best one, at least with the *Rejected* and *CBP*$_{30}$ functions, which means that perhaps it could be interesting to dynamically adapt the monitoring period as a function of the call arrival times. This happens because the *Rejected* and *CBP* functions depend on the number of rejected connections during the monitoring periods and if they are very short then, depending on the offered load, it is unlikely that there would be enough rejections to provoke a congestion detection.

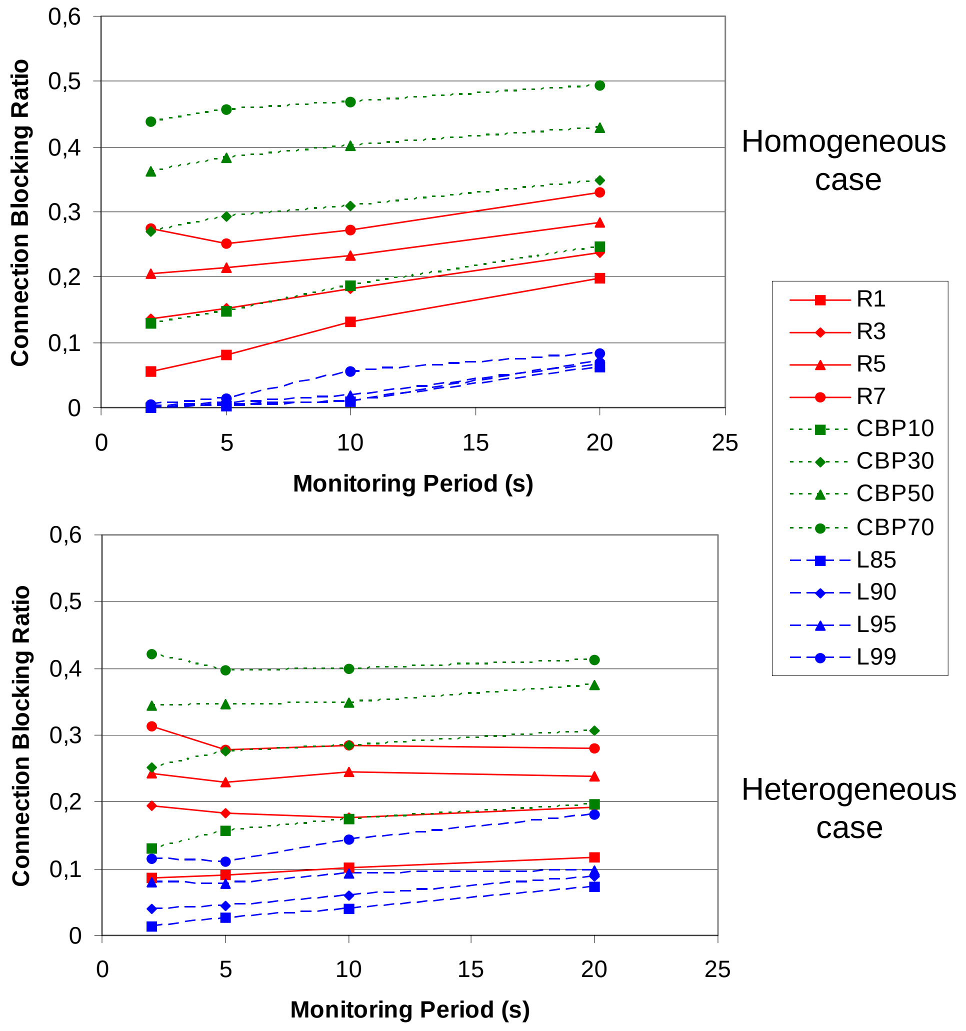


Fig. 5.2: Connection Blocking Ratio in relation to Monitoring Period for all the TFs and their Limits.

It can also be seen that in the heterogeneous case the monitoring period has less affect than in the homogeneous case. This again indicates that the monitoring period is not a crucial parameter within the lightweight monitoring proposed. The most important issue, that can also be seen in other figures, is the selection of the TF, and secondly the selection of the limit value. According to the results the best TF is the *Load* (L), followed by the *Rejected* (R) and finally the *CBP*. Figure 5.3 shows the differences between the TFs and their limits. The lower the limits the better the TF's behaviour for all three TFs, and in this case, there are no important differences between the homogeneous and heterogeneous connection cases. Again, the monitoring period is not a crucial parameter.

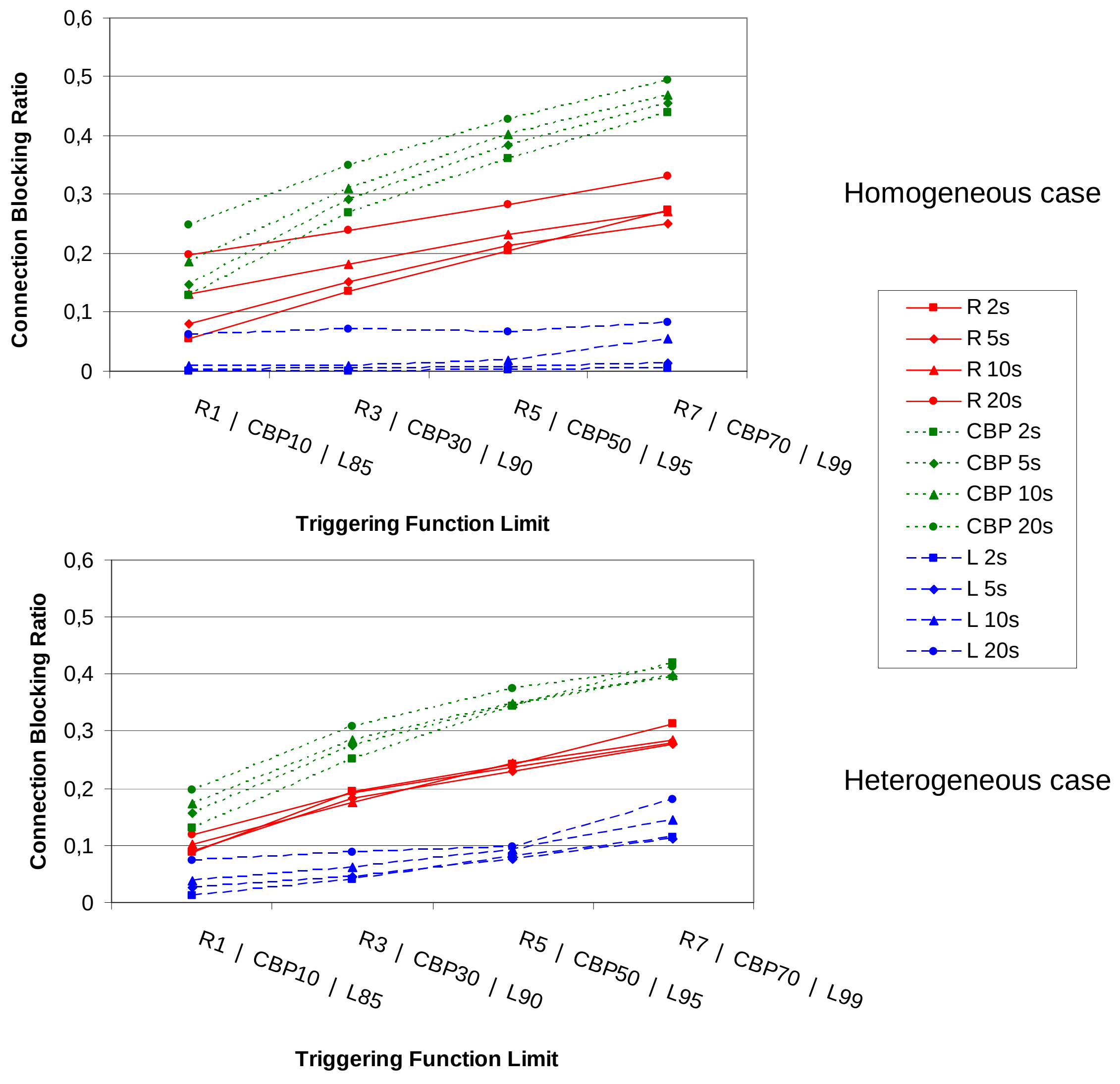


Fig. 5.3: Connection Blocking Ratio in relation to TF Limits for all the TFs and the Monitoring Periods.

TF Behaviour in relation to the Step Size

After the comparison of the TFs regarding the monitoring periods, a similar comparison regarding the step sizes was carried out using four different step sizes (Table 5.2). It is clear that the greater the step size the better the TF behaviour is (smaller connection blocking ratio). The main reason is that the greater the step size the longer the time the LP can get congested again and reject user connections. Figure 5.4 shows the connection blocking ratio versus the step size for all the TFs and their limits. Figure 5.5 shows the same ratio versus the different TFs' limits for all the TFs and the different step sizes used. Both figures shows that with the selected limits the best TF is again the Load, followed by the Rejected and finally the CBP functions.

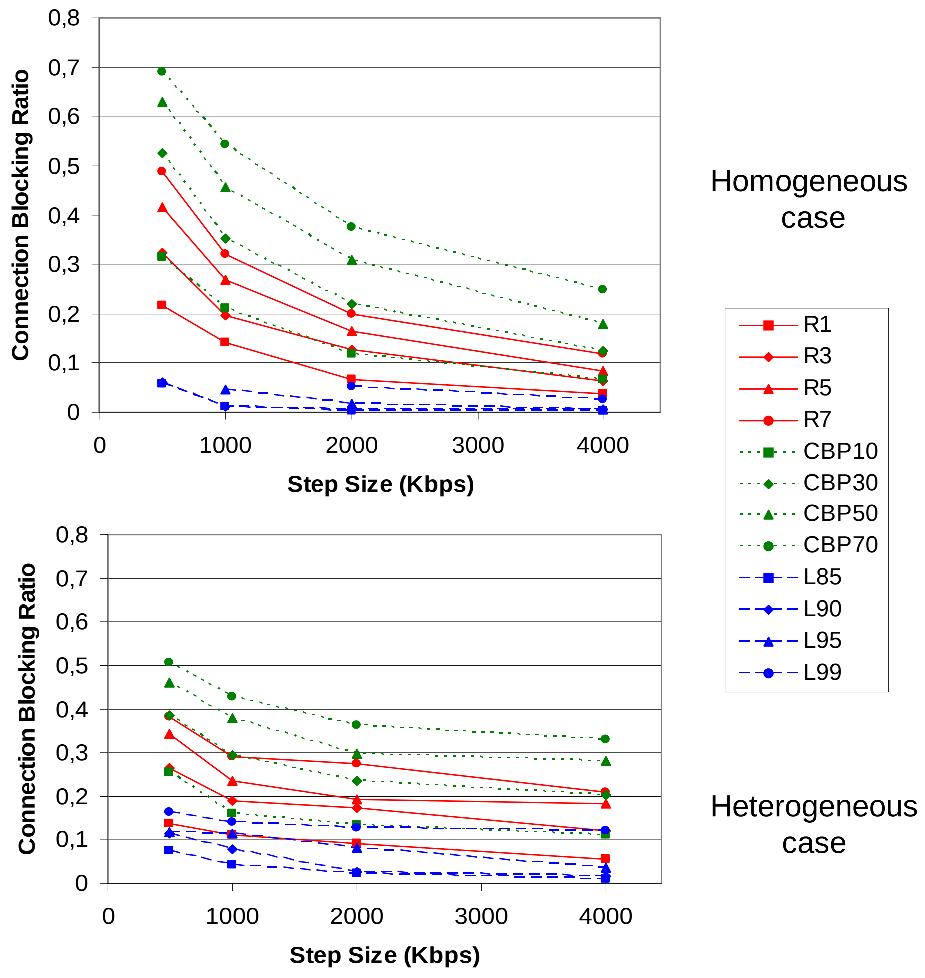


Fig. 5.4: Connection Blocking Ratio in relation to Step Size for all the TFs and their Limits.

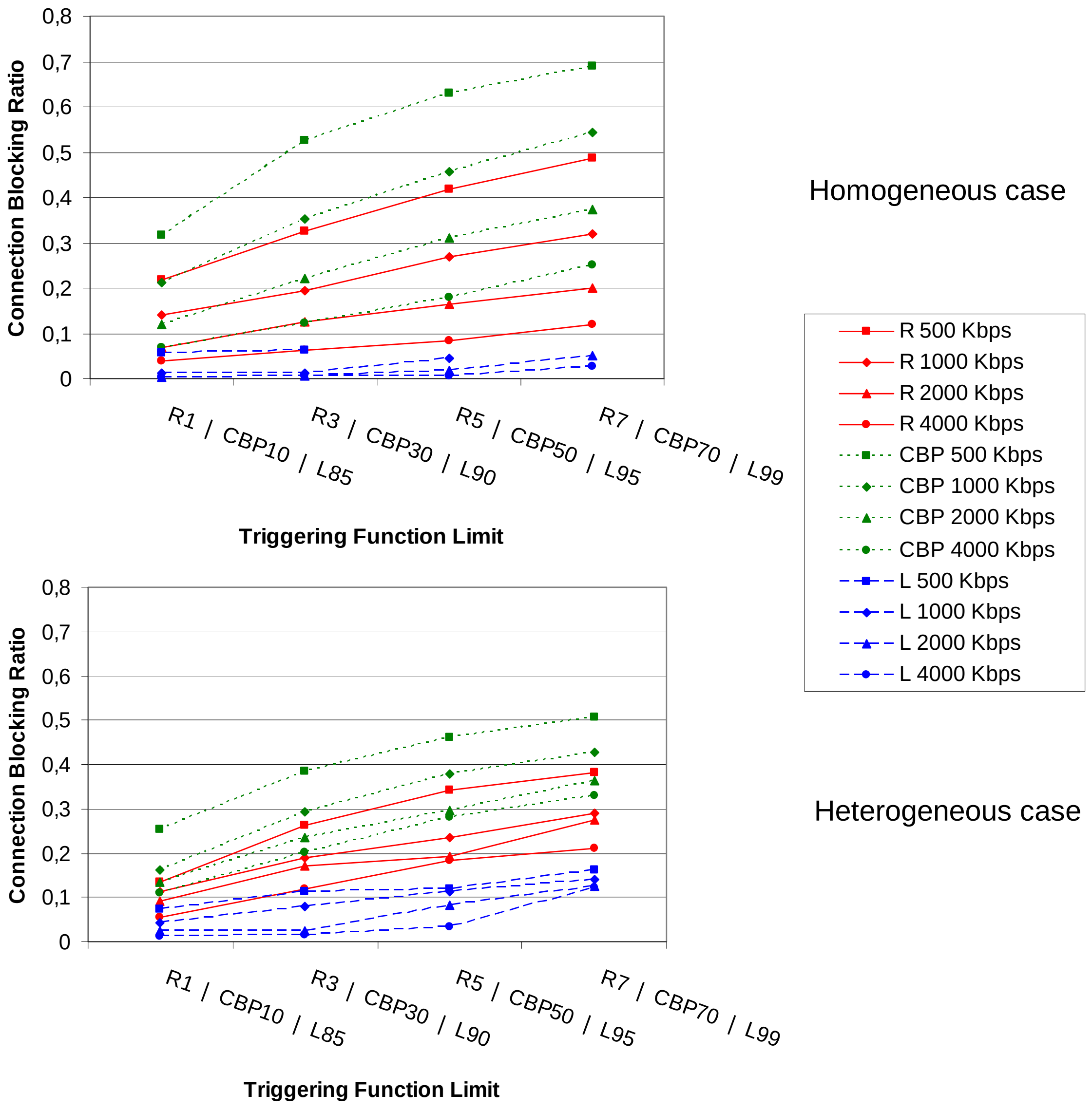


Fig. 5.5: Connection Blocking Ratio in relation to TF Limits for all the TFs and the Step Sizes.

Monitoring and Congestion Detection. Discussion and Conclusions.

The main conclusion we can make from these results is that the Load function has the best performance in most of the situations. This is because this function is preventive and requests an LP increase before there are rejections in the LPs. This means that rejections are only produced when the monitoring time is long enough. The best performances are obtained by Load 85%, Load 90%, and Load 95%, while Load 99% gives good but similar results compared to Rejected 1 and CBP 10%, especially in the heterogeneous case. However the Load function has a severe drawback: the misuse of the 15%, 10% or 5% of the LP capacity in the cases of Load 85%, Load 90%, and Load 95% respectively, which are not usually used. This could prove to be unacceptable, especially with large LPs or with a great number of LPs, in which case a significant network capacity could become unusable.

From the analysis of these combinations of TFs, their limits, the monitoring period, and the step size, the use of several of the TFs at the same time seems an interesting proposal. Depending on the traffic type, e.g. high priority or very sensitive users or services, the use of a Load TF could be interesting, while in other cases other TFs could be used, or even several LPs could be left fixed, i.e. unmonitored. This is possible because of the complete independence of every M agent, where each one can be configured accordingly. Therefore each LP can be monitored using different TFs and limits.

The dynamic adaptation of the monitoring period is also possible, for instance depending on the load of the LP. If the LP is not carrying many user connections and its capacity is being underused then fast monitoring is not necessary and the period could be augmented saving processing time and monitoring actions. On the contrary, while the LP becomes near congestion it could be interesting to reduce the monitoring period to control the LP better. The monitoring period can also be adapted depending on the arrival time distribution of the user connections, or the traffic characteristics.

The step size can also be adjusted dynamically, depending on the type of the connections (the step size should be at least the size of one user connection) and the free bandwidth on the physical link (it is worthless trying to increase an LP using a big step size if there is less free bandwidth in the physical link) or the status of other LPs going through the same physical link (if most of the other LPs are near congestion it is likely that several of them could ask for a bandwidth increase). Regarding the step size, it is better to use big sizes. However, the use of big step sizes could provoke a large number of LP decreases (pre-emption) and thus increase the number of messages between P agents.

There is also the possibility for the network administrator or other management systems to define, in the establishment time, minimum and maximum capacity values for some LPs in order to limit the extremes of the dynamic management in the network.

In a heterogeneous connection scenario with a large difference between the connection sizes it could occur that the small connections are being accepted while the large connections are not. This could result in an unfair situation if there is not enough combined rejection to be detected by the TF to increase the LP. This could be solved in two different ways. First the proposed dynamic bandwidth management system could use different monitoring TFs and parameters for different connection types in the same LP, and increase the LP when one or the other TF detects congestion. Another possibility is to leave the proposed dynamic bandwidth management system as is and leave the fairness control to the CAC mechanism.

The objective of the M agents and the Triggering Functions is to perform lightweight monitoring in order to detect persistent congested situations and activate the bandwidth readjustment mechanisms. This is achieved by the proposed lightweight M agents and simple TFs that can perform fast distributed but not real-time monitoring, which does not overwhelm the network elements.

5.3.2 Bandwidth Reallocation

This sub-section describes the experiments realised in order to verify that the different bandwidth reallocation mechanisms proposed in chapter 4 are operating correctly. More specifically it shows the results of the mechanisms used by the P agents in order to increase the bandwidth of a congested LP, they are the following: Free-Bandwidth-Only (FBO), First-Node-Only (FNO), and Any-LP (ALP). Three experiments are presented, which were performed using different network topologies and traffic conditions.

Operation Test of the Bandwidth Reallocation Mechanisms

The first experiment consists in two simple situations depicted in Figure 5.6. There are two networks both with two unidirectional LPs, and in both cases LP1 is not congested (mean interarrival time = 4 s and mean duration time =6 s) and LP2 is congested (mean interarrival time = 1 s and mean duration time = 10 s). Initially all the LPs have a capacity of 8 Mbps each and the physical links are of 16 Mbps each. The offered user connections are 2 Mbps each and the step size for the increment/decrement of the bandwidth is also 2 Mbps. The Rejected TF was also selected with a limit of 5.

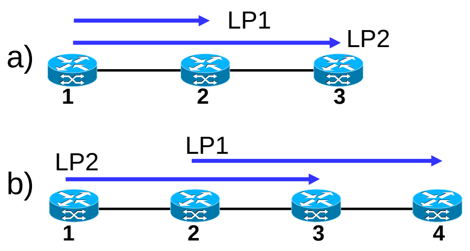


Fig. 5.6: Scenario 2 – Bandwidth Reallocation Mechanisms Operation Test.

Figure 5.7 shows the results for case a). In this case the FNO and ALP algorithms perform equally as expected, because both act by reducing the bandwidth of LP1 and the ALP has no advantage over the FNO, while the FBO algorithm performs badly because there is no free bandwidth in link 1-2, therefore it is not capable of performing any change and the network remains static. Figure 5.8 shows the results for situation b). In this case, the ALP algorithm takes advantage of the unused bandwidth of any other LP going through the same links as the congested LP2. However, the FBO cannot use the free bandwidth in the physical links because there is no free bandwidth in link 2-3 and the FNO cannot use the unused bandwidth assigned to LP1 because it does not have the same origin as the congested LP.

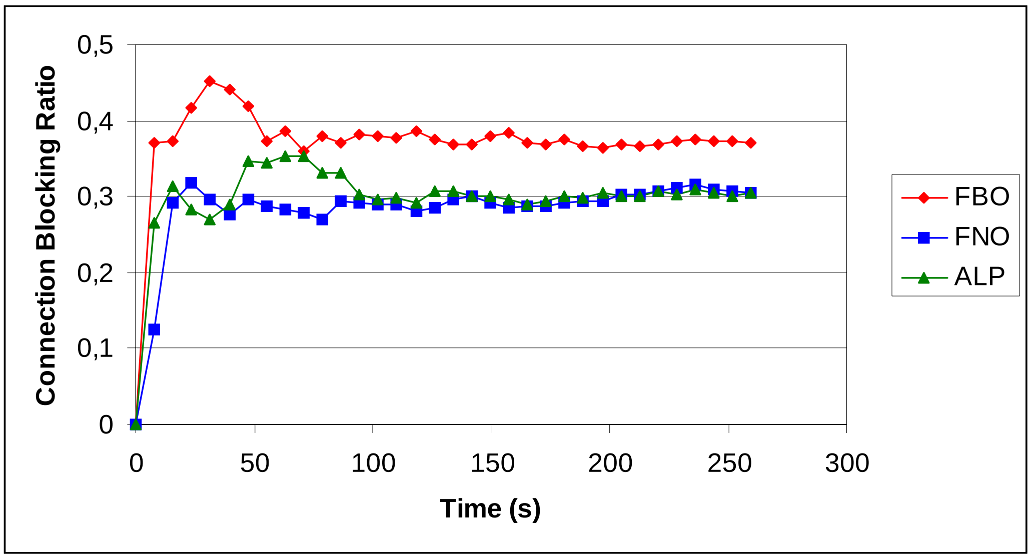


Fig. 5.7: Connection Blocking Ratio versus Time in Scenario 2-a.

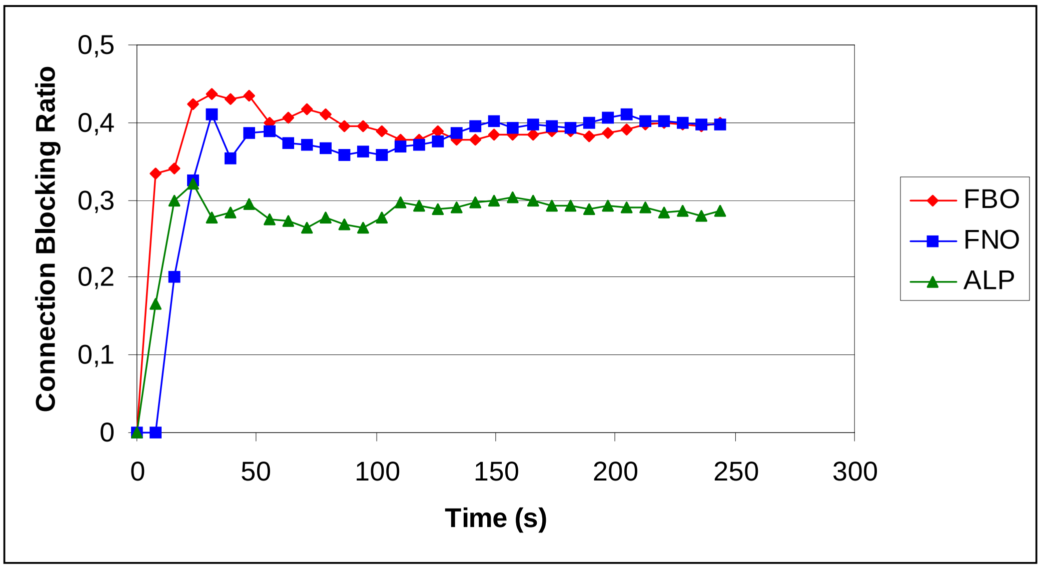


Fig. 5.8: Connection Blocking Ratio versus Time in Scenario 2-b.

Behaviour Test 1 of the Bandwidth Reallocation Mechanisms

The bandwidth reallocation mechanisms performed as expected in the previous simple tests. After the initial tests the objective was to use bigger networks and in more complex scenarios in order to evaluate the behaviour of the algorithms. We started with scenario 3 first, which consists in a full-meshed 5-node logical network established over a 9-node physical network (there are 4 core nodes and 5 edge nodes) depicted in Figure 5.9. All the physical links are bi-directional with capacities of 100 Mbps between edge nodes and between edge and core nodes, and with capacities of 200 Mbps between core nodes. There are 20 established unidirectional LPs, all of them initially with an assigned bandwidth of 10Mbps which defines the full meshed network between the 5 edge nodes.

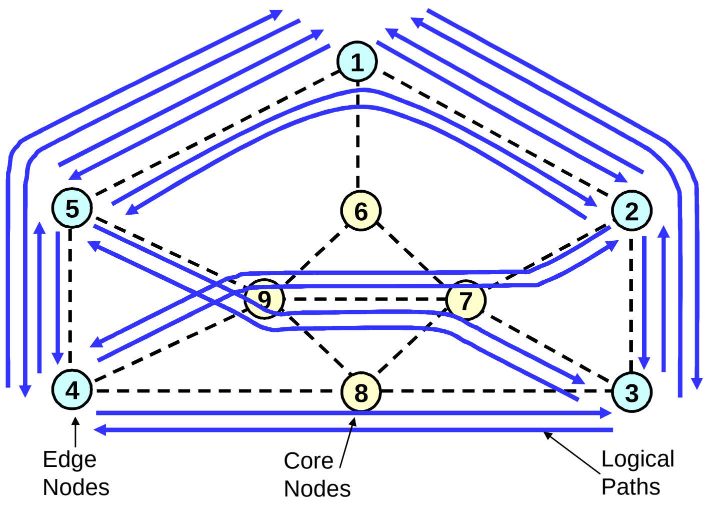


Fig. 5.9: Scenario 3 – Bandwidth Reallocation Mechanisms Behaviour Test 1.

The TF was fixed to Rejected with a limit of 5, and the step size was fixed to 2 Mbps. In order to provoke congestion and the action of the bandwidth reallocation mechanisms we chose an identical mean offered load of about 100 Mbps per each LP including connections of all the five classes presented in Table 5.1. This load cannot be absorbed by the physical network itself thus a high call blocking ratio is expected. The objective is that the initial LPs need to be adapted to the offered load by the bandwidth reallocation mechanisms. However, we found little differences in the results (Table 5.4) between the three methods due to a combination of two reasons: the static and equal offered load for all the LPs means that once the logical network is adapted by the mechanisms then it remains almost static and no more adjustments are produced; the high offered load for all the LPs means that after the initial adaptation of the logical network there is no free bandwidth on the links but also no underused LPs, therefore all the mechanisms perform in a similar way.

To avoid this situation and take advantage of the bandwidth reallocation mechanisms, it is necessary that the offered load for each LP changes during the simulation. For this reason, the LPs have been divided into two groups and the simulation time into three periods. In the first and the last period one group of LPs has a high offered load and the other group a low one, while in the second period the group of LPs initially with a high load then have a low load and vice versa. In each period the reallocation mechanisms must make changes to adapt the logical network to the new offered load. The results are also depicted in Table 5.4. In this case there is a significant difference between the three bandwidth reallocation algorithms. As expected ALP shows the best results followed by FNO, and finally the FBO algorithm.

Bandwidth Reallocation Algorithm	*Mean CBP (static offered load)*	*Mean CBP (changing offered load)*
FBO	0.681	0.703
FNO	0.670	0.654
ALP	0.676	0.619

Table 5.4: Bandwidth Reallocation Results for the Scenario 3

Behaviour Test 2 of the Bandwidth Reallocation Mechanisms

Finally we performed an experiment similar to the previous one using a different scenario presented in Figure 5.10. In this case, there are 15 nodes of which 7 are edge nodes and 8 are considered core network nodes. All links are bi-directional with a capacity of 166Mbps. There are 24 unidirectional LPs with an initial capacity of 20Mbps each. Half of them are underused (mean interarrival time is 2 s and mean duration time is 7 s) and the other half are congested (mean interarrival time 1 s and mean duration time 170 s). Every 12 minutes the traffic is switched and the underused LPs become congested and vice versa in a similar way to scenario 3 changing offered load.

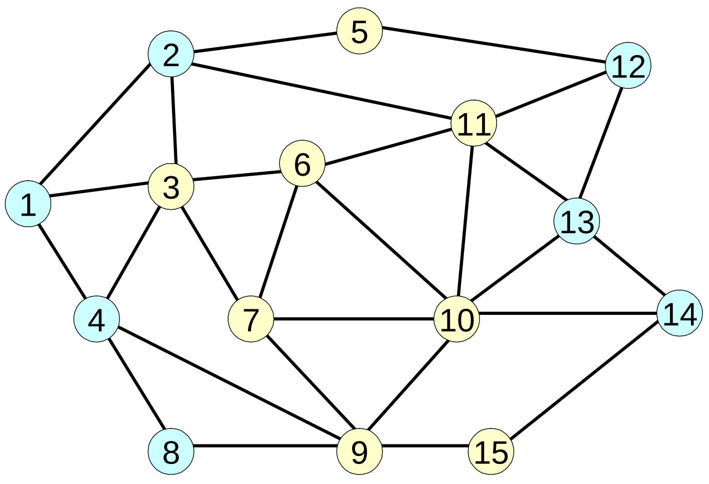


Fig. 5.10: Scenario 4 – Bandwidth Reallocation Mechanisms Behaviour Test 2.

Figure 5.11 shows the behaviour of the three bandwidth reallocation algorithms over 50 minutes of simulation (approx 42000 connection requests). The algorithm ALP adapts better to the load changes and maintains a lower CBP. However the number of messages between P agents, summarised in Table 5.5, of the ALP algorithm is much higher than the other algorithms. Although these messages are spread over the network and simulation time, a low number of messages is desirable in order to maintain good scalability, and in some logical network topologies perhaps it could also be interesting to use the FNO algorithm.

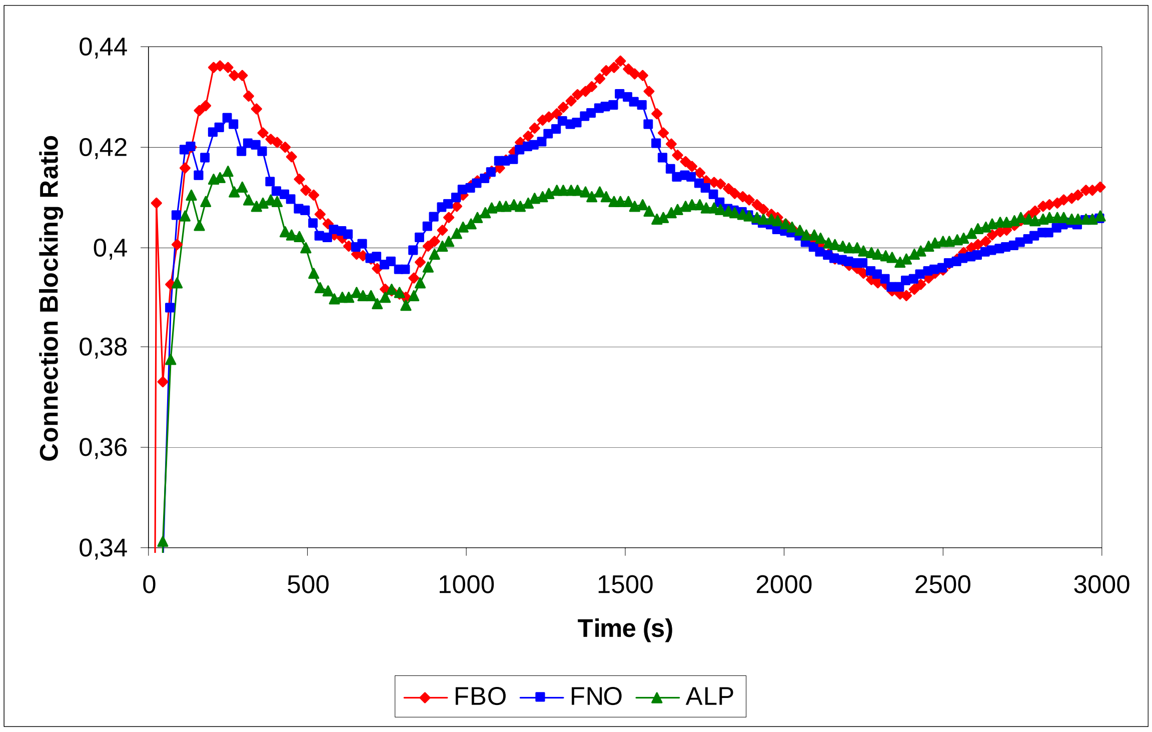


Fig. 5.11: Connection Blocking Ratio versus Time in Scenario 4.

Bandwidth Reallocation Algorithm	*Number of Messages*
FBO	5478
FNO	13339
ALP	21402

Table 5.5: Number of Messages for Scenario 4.

5.3.3 Logical Path Rerouting

The next set of experiments has been carried out with the purpose of verifying that the proposed LP rerouting mechanism is operating correctly. More specifically, an experiment is presented with the objective of evaluating how the weights ω1 and ω2 of Formula 4.9 act, and a second experiment comparing the behaviour when using a bandwidth reallocation algorithm with and without the rerouting option in the same scenario with the same offered load.

Weight Influence Test of the LP Rerouting Mechanism

We started be carrying out a simple test with the objective of trying different values of the weights ω1 and ω2 in Formula 4.9 (presented here again). For this experiment we used the physical network and LPs depicted in Figure 5.12. Note that Formula 4.9 is used to order the paths that have enough capacity to reroute the congested LP and make a choice. That formula does not depend on the size of the LP to be rerouted, it is just to decide if it is better to select the shortest path or the path with more free bandwidth. The absolute values of ω1 and ω2 do not matter, only their ratio.

$$\varphi_R = \omega_1 \cdot \frac{1}{H} + \omega_2 \cdot \frac{\sum_{i=0}^{H-1} (H-i)\cdot p_i}{\max(p_i)\cdot \sum_{i=0}^{H-1}(H-i)} \qquad \text{(Duplicate of the Formula 4.9)}$$

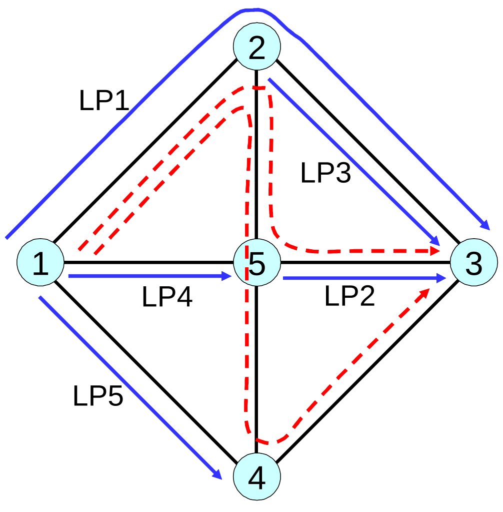


Fig. 5.12: Scenario 5 – Rerouting Mechanism Weight Influence Test.

Focussing on the example, let us suppose that all the physical links are of 100 bandwidth units. Suppose then that LP3, LP4, and LP5 have a bandwidth of 60 units and that LP1 has a bandwidth of 35 units and it is congested. LP2 has a bandwidth less than or equal to 50 units. The aim is to increase LP1 by 10 units, i.e. to 45, so of all the possible routes from node 1 to node 3 there are only two possibilities (the broken red lines in Figure 5.12): route R1 = 1-2-5-3 and route R2 = 1-2-5-4-3. Note that R1 is shorter but there is the LP2 bandwidth that may affect the decision. The second possibility, R2, has the maximum bandwidth (100 units) available in all the links it traverses. Therefore the decision also depends on the selected ω1 and ω2 and the bandwidth of LP2.

With the help of a spreadsheet the critical values of the LP2 bandwidth have been calculated for several values of ω1 and ω2 and simulation tests have been used to verify that the expected results have been obtained. Some of the tested values are depicted in Table 5.6, which shows the values of the LP2 bandwidth that make the P-agents choice one route or the other for several values of ω1 and ω2. The results indicate that ω2 should be greater than or equal to ω1, otherwise in a network with many LPs and a considerable percentage of the bandwidth in the links occupied the shortest of the possible routes will always be selected.

Value of ω_1 (ω_2 fixed to 2)	*Route R1 selected when*	*Route R2 selected when*
0.5	$0 \leq LP2 < 13$	$13 \leq LP2 \leq 50$
1	$0 \leq LP2 < 25$	$25 \leq LP2 \leq 50$
1.5	$0 \leq LP2 < 38$	$38 \leq LP2 \leq 50$
2	$0 \leq LP2 < 50$	50

Table 5.6: Comparison of ω_1 and ω_2 Values in Scenario 5.

Bandwidth Reallocation with and without LP Rerouting Comparative

In this experiment we wish to show in which situations it could be an interesting advantage to use the LP Rerouting mechanisms in the case that the Bandwidth Reallocation mechanisms cannot adjust the logical network better. In the proposed situation the physical links are not loaded at 100% of their capacity and, after the logical network has been adjusted to the offered traffic, there is an increase in the offered connections of several of the LPs. This means that several LPs are rerouted through less congested links. The idea is to compare two identical situations one with rerouting and the other without that possibility.

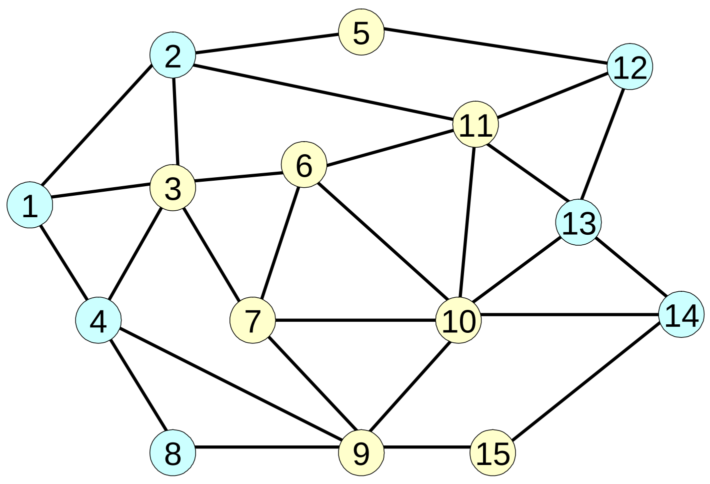


Fig. 5.13: Scenario 6 – LP Rerouting Mechanism Test.

In this scenario the network depicted in Figure 5.13 is used with all the physical links having a capacity of 166 Mbps. The established LPs (34) are listed in Table 5.7, and are established between edge nodes, they all originally have 20 Mbps of assigned capacity. The simulations last for one hour (3600 s). The bandwidth reallocation algorithm (ALP) initially has about 1000 s to adapt the original LPs to the initial offered load. After that, the offered load of a few LPs is changed around the time of 1000 s and the offered load of another few LPs around the time of 2000s (it changes in both directions). The numbers of the offered loads in Table 5.7 express the Mean Interarrival Time (s) / the Mean Duration Time (s) / the Traffic Class (listed in Table 5.1).

LP (one in each direction)	*Initial Offered Load*	*New Offered Load*	*Change Time (s)*
1-2	3/20/2		
1-4	3/20/2		
1-3-6-11-12	2/30/2	1/60/2 + 1/60/3 + 2/60/4	2000
1-3-6-11-13	2/20/2		
1-3-6-10-14	3/20/2		
2-5-12	3/20/2		
2-11-13	2/20/2	1/60/2 + 1/60/3 + 2/60/4	2100
2-3-7-10-14	3/20/2		
4-8	3/20/2		
4-3-6-11-12	2/20/2	1/60/2 + 1/60/3 + 2/60/4	1000
4-9-10-13	2/30/2		
4-9-15-14	2/20/2	1/60/2 + 1/60/3 + 2/60/4	1100
8-9-10-13	2/20/2	1/60/2 + 1/60/3 + 2/60/4	1200
8-9-10-11-12	2/30/2		
8-9-15-14	2/30/2		
12-13	3/20/2		
13-14	3/20/2		

Table 5.7: LPs and Offered Loads for Scenario 6.

The selected bandwidth reallocation algorithm is the ALP and the TF is the Rejected with a limit of 5. The monitoring time was fixed at 5 seconds and the step size fixed at 4 Mbps. The experiment was carried out several times without rerouting and several times with rerouting. In the case of rerouting there were between 4 and 7 LPs rerouted at the end of the simulation time. There are many LPs that do not change their offered load and the offered load changes occur at around 1/3 and 2/3 of the way through the simulation, this means that the cases of rerouted LPs are in the second half of the simulation. This fact, along with the offered load not being extremely high as in other experiments means that there is a small but significant difference between the rerouting case and the case without rerouting. As a consequence, there is a connection blocking ratio less than 0.1, which is different from other experiments with connection blocking ratios of 0.3 and 0.4. In order to present the results, the LPs have been grouped according to their origin node and the rejection ratio per node is presented (Figure 5.14). From this point of view node 1 and node 12 are benefited most by the rerouting mechanism.

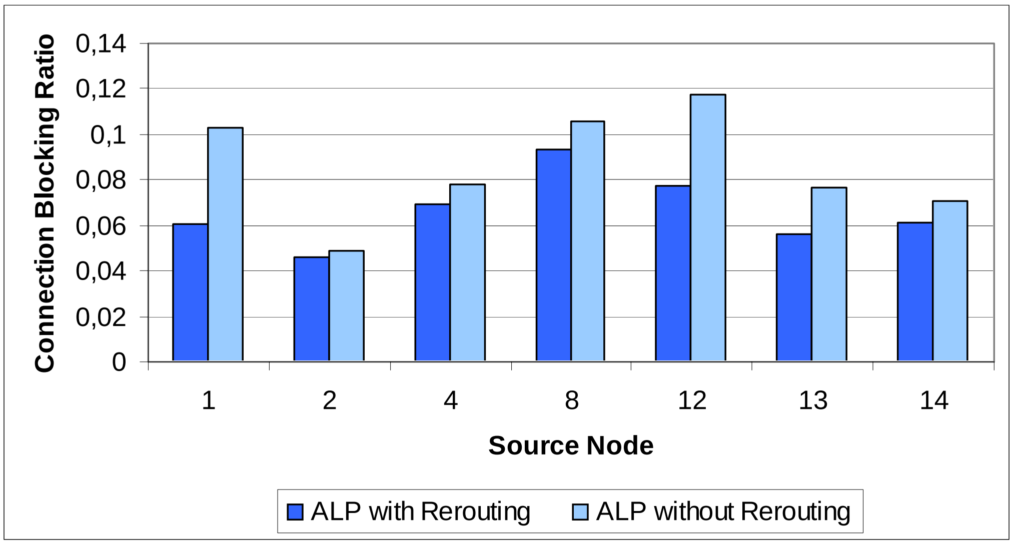


Fig. 5.14: Connection Blocking Ratio versus Node Number.

The selected weights were ω1 = 1 and ω2 =1, and because several links had only one LP with a small load the shortest path was not always selected. Another point was that two initially symmetrical LPs, in opposite directions, could be rerouted through asymmetrical new routes or even the case that one is rerouted but the other one in the opposite direction is not. This can happen because we have not forced both directions between the same two nodes to go through the same route or have the same bandwidth assigned to them, they are completely independent.

Regarding the messages sent by the P-agents, it seems that there should have been more messages in the rerouting case than in the case without rerouting. However, the contrary happened because when a LP is congested and there is no possibility of increasing it using the ALP bandwidth management algorithm, the P-agents keep trying and sending messages in order to find an LP that can reduce its capacity. Moreover, in the rerouting case when an LP is in the process of being rerouted, it is blocked and the possible congestion alarms are discarded, which also helps to save messages. The number of messages is summarised in the Table 5.8.

Case	*Total number of messages*	*Maximum number of messages sent by a node*	*Minimum number of messages sent by a node*
ALP with Rerouting	1858	274	23
ALP without Rerouting	3814	758	3

Table 5.8: Number of Messages for the Rerouting Scenario 6.

5.3.4 Restoration Mechanism and Spare Capacity Management

This sub-section describes the tests we have carried out in order to verify that the restoration mechanisms and the spare capacity management are operating correctly. However, in this case we do not propose new algorithms as in the previous sections and we have only adapted a pre-planned restoration mechanism based on the use of end-to-end backup LPs to the proposed MAS. An LP and its backup must go through disjoint paths. In the case that the desired protection level is defined, for instance against a single link failure, then the reserved bandwidth for the LPs can be shared between different backups in order to save bandwidth. Two different experiments are presented in order to see how the proposed MAS performs when a fault occurs. The first one aims to measure the restoration time and the second one to check that the restoration mechanism is operating properly when the fault affects several LPs.

Response Time Evaluation of the Restoration Mechanism

This experiment is to measure the response time of the proposed MAS, two different times have been measured:

- The first one is from when the network node sends an alarm message to the P-agent responsible for that node indicating that a physical link has just failed, until when the traffic of an affected LP is switched to the backup LP and the required bandwidth of the backup LP is captured. This includes the time the message *LinkFailed* takes to go upstream to the origin node, and the time the message *CaptureBW* takes to go downstream the backup LP to the destination node.
- The second measured time corresponds to the reparation process when the user connections are switched back to the original LP. This measures the time from the node origin of the failed link to send the *LinkRepaired* notification to the responsible P-agent, and the time for this message to be sent upstream to the origin node of an affected LP where the switch back is performed.

The tests have been carried out using a single computer (Sun Enterprise 250, single 64-bit 300 MHz processor UltraSPARC-II, 384 Mb of RAM). The measured times include the transmission, queuing, and processing times of the messages in the P-agents, while the propagation time should be considered depreciable as the processes are on the same computer. However, in these experiments the real time is measured, not simulated time, so it is also affected by the load of the computer. This can also give us an idea of the load of the P-agents and the simulator. For instance, using a small network with 4 nodes, this means that there are 4 processes for the simulation of the 4 nodes (implemented in C++) and another 4 processes for the P-agents (implemented in Java). There are also, in this case, the 2 processes for the event generators (connections and faults). In this case the achieved times are in the order of hundreds of milliseconds.

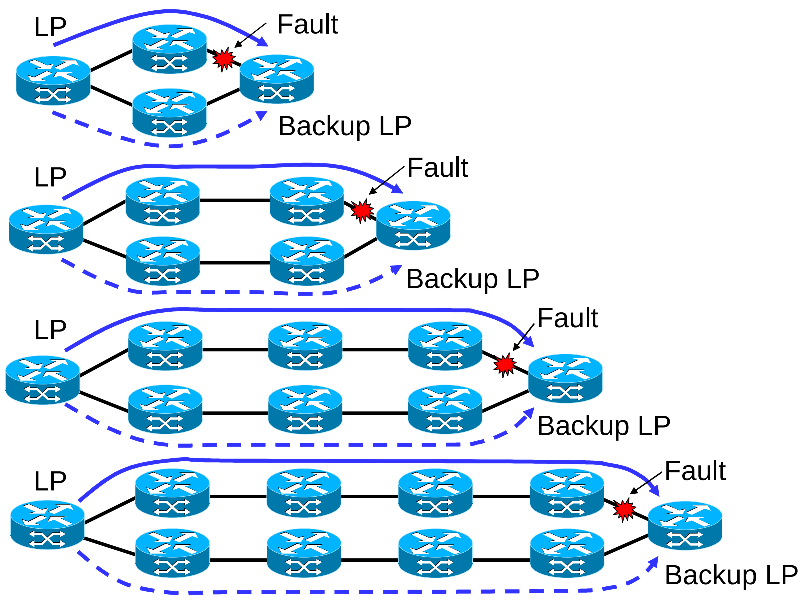


Fig 5.15: Scenario 7 – Ring Topologies for the Restoration Experiments

Figure 5.15 shows the different networks used in this experiment, all of them with a ring topology. There is a 4, 6, 8, and 10 node ring where an LP and its corresponding backup between two opposite nodes are established. Figure 5.16 displays the time from the fault detection until the bandwidth of the backup is captured and the user connections are switched over to the backup LP. Figure 5.17 displays the time from the link restoration until the user connections are switched back to the original LP. In both cases we have calculated a 95% confidence interval which is represented in the figures.

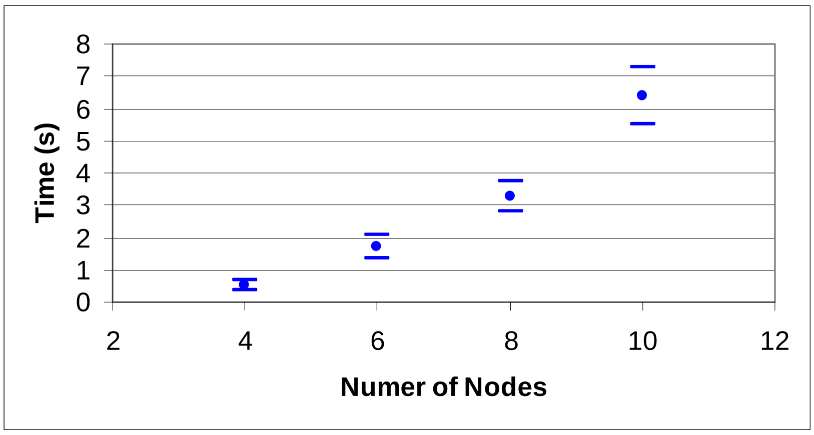


Fig 5.16: Time from the Fault Detection until the Backup LP Bandwidth is Captured

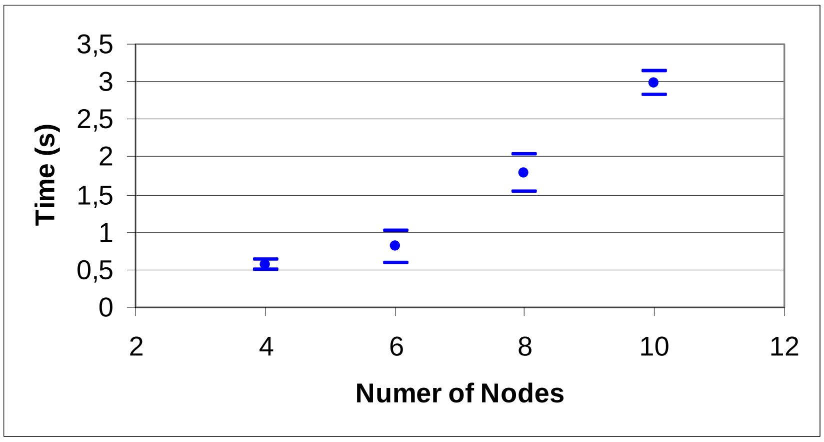


Fig 5.17: Time from the Link Restoration until the User Connections are Switched Back to the Original LP.

The first measured time is approximately the double of the second because the first case is comprised of two messages going up and downstream while the second is comprised of only one message going upstream. The time increases when more nodes are added, which is also to be expected, however it is not possible to distinguish which part of this increment is due to the processing time of the P-agents and which part is due to the load increment of the computer.

This experiment was also useful in order to verify that the restoration mechanisms were operating properly. The P-agents detect both the failures and link-repaired indications from the nodes and the restoration messages are also used as defined. The switchover mechanism is performed by the P-agent by sending a command to the node which makes a substitution in the connection tables.

Multiple LP Restoration and Spare Capacity Management Test

The second experiment was designed in order to check that the spare capacity management operates correctly when the desired protection level is against one single link failure, and also that the backup LPs are activated correctly when a link failure affects more than one LP. In this case, a bigger network (Figure 5.18) is simulated. Note that in this experiment three computers were used.

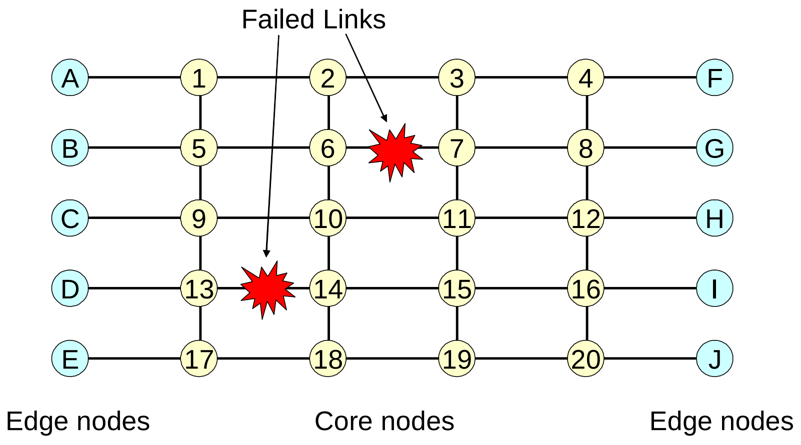


Fig. 5.18: Scenario 8 – Network Topology for the Spare Capacity Management Scenario.

There are 30 nodes of which 10 represent edge nodes. There are 50 established LPs connecting every edge node from the left side (A to E) to all the nodes on the right side (F to J) and vice versa. There are also 50 backup LPs protecting the working ones, which go through disjoint paths of their respective working LPs. The logical network planning is out of the scope of this work, and the paths were established from an initialisation table. However, in the initialisation process, the P-agents reserved the whole required bandwidth for the working LPs, while the backup LPs shared their bandwidth applying the criteria defined in the previous chapter.

This experiment simply consists in provoking two link failures, which are not simultaneous, and verifying that the bandwidth required for the backups is captured accordingly from the bandwidth reserved by them. The first fault to occur was in link 6-7. The P-agent at node 6 detected the fault after receiving an alarm from the node and sent *LinkFailed* messages to every edge node that had an LP going through link 6-7. Four LPs were affected, therefore when the origin nodes received the messages they activated the backups and switched the established user connections from the original LPs to the backup ones. When the fault was corrected, link 6-7 was restored and a similar mechanism switched back the user connections to the original LPs. After a small delay link 13-14 failed and the mechanisms acted in the same way.

Conclusions

With both these experiments we have verified that the implemented restoration mechanisms of the proposed MAS are operating correctly. We think that the proposed MAS could take charge of these mechanisms in a real network performing fast enough. However, it is also possible to have hybrid situations where, for instance all the restoration processes are performed at a lower level and only the management of the spare bandwidth and the co-ordination with the bandwidth management mechanisms is performed by the proposed MAS. In this case notification messages from those lower level protection mechanisms are necessary to make the agents aware of the faults and the active backup LPs.

5.4 Putting Everything Together

In the previous experiments we have tested the Triggering Functions and their parameters, the bandwidth management mechanisms, and the restoration mechanisms including the spare bandwidth management. However each time the experiments were focused on a single mechanism. In many cases several of the proposed mechanisms were used together and therefore, all the mechanisms have already been tested together in the previous experiments. However, this section describes one more experiment, which is very similar to the second rerouting experiment (section 5.3.3) where we already used several mechanisms together, but on this occasion, the objective is to focus on the coordination of the mechanisms rather than on a single mechanism or on the obtained network performance.

For the coordination of all the mechanisms and to avoid interferences between them, we applied the rules and constraints proposed in section 4.4. More specifically,

- The working LP and the backup LP should be increased and decreased accordingly. The decreasing procedure is more or less straight, however in the increasing process the proposed MAS tries to increase both the working LP and the backup LP simultaneously. If one of them or both fail, then the increasing process is aborted.
- For the rerouting process we simply decided that only the working LP can be rerouted, and in that case, the possible routes that coincide with the backup LP in some links are simply eliminated from the list and thus cannot be selected. The possibility of rerouting backup LPs in order to free bandwidth in the physical links is not contemplated in the experiments carried out, this is left for future work.
- When a failure occurs the bandwidth of the affected LPs and their corresponding backups cannot be increased nor decreased, and the possible congestion alarms are ignored by the P-agent.

Taking all this into account, an experiment was carried out using the physical topology depicted in Figure 5.19, and the same logical network (described in Table 5.9). However the offered load of all the LPs was reduced to approximately half of the values described in Table 5.7. On this occasion backup LPs for all the LPs in the network (also specified in Table 5.9) were established, and the simulation was carried out using the bandwidth reallocation algorithm ALP along with the LP rerouting mechanism, the only differences being that in this case there is some amount of bandwidth in the links reserved for the backup LPs and a lower offered load.

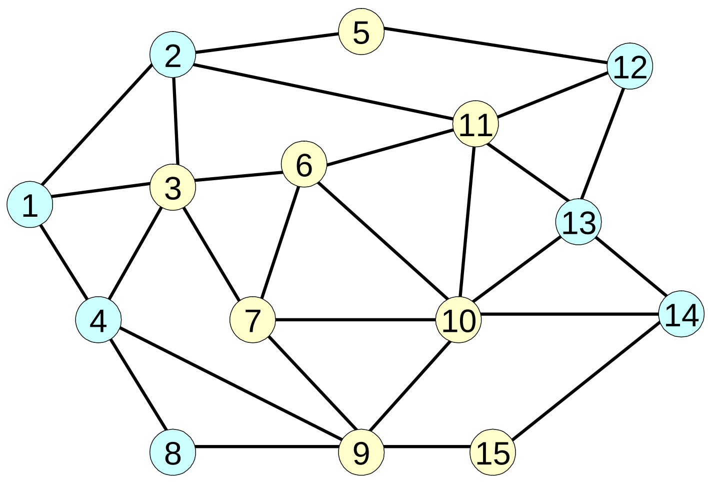


Fig. 5.19: Scenario 9 – Test for Putting Everything Together.

LP (one in each direction)	*Backup LP (one in each direction)*
1-2	1-3-2
1-4	1-3-4
1-3-6-11-12	1-2-5-12
1-3-6-11-13	1-4-9-10-13
1-3-6-10-14	1-2-11-13-14
2-5-12	2-11-12
2-11-13	2-5-12-13
2-3-7-10-14	2-11-13-14
4-8	4-9-8
4-3-6-11-12	4-9-10-13-12
4-9-10-13	4-3-6-11-13
4-9-15-14	4-3-7-10-14
8-9-10-13	8-4-3-6-11-13
8-9-10-11-12	8-4-3-2-5-12
8-9-15-14	8-4-3-7-10-14
12-13	12-11-13
13-14	13-10-14

Table 5.9: LPs and Backup LPs.

In the adaptation of the logical network to the offered load, usually both the working LP and its backup can be simultaneously increased. In some cases there is no need to reserve more bandwidth in several links for a backup increase because the spare bandwidth can be shared. There is no difference in the rerouting process because only the working paths can be rerouted, however they have fewer routes to select.

A link failure was also simulated. The failed link was 6-11 and this required the activation of three backup LPs, and the switchover of the connections from the working LPs to their backups. The failure lasted for a short time (200 s) and then the link was restored. After that, the user connections where switched back again to the original working LP. During the failure the affected LPs were fixed and the congestion alarms, and petitions for an increase or decrease of their bandwidth were ignored. Meanwhile, the other LPs not affected by the failure were managed by the P-agents as usual.

Conclusions

P-agents, with the help of M-agents, manage the logical network successfully and coordinate the different mechanisms appropriately. The coordination constraints and rules worked properly and the view of the node status that each P-agent maintains did not become inconsistent at any moment.

5.4.1 Scalability Study

One of the main objectives of the proposed MAS is to achieve good scalability. In this section we present a mathematical model of the interactions of the software agents in our system. The purpose is to evaluate the system's scalability based on the ideas presented in section 3.7.1. The proposed MAS exhibits suitable scalability for a wide range of scales.

In this study, the network administrator is considered to be the user of the proposed dynamic network resource management system. From this point of view, the mathematical model is developed from the starting point given by Formula 3.1. Function *F* is also assumed to be the one suggested in [Jogalekar and Woodside, 1998], presented in Formula 5.2.

$$\Psi_P = \frac{F(\lambda_2, QoS_2, C_2)}{F(\lambda_1, QoS_1, C_1)}$$

(Duplicate of the Formula 3.1)

$$F = \frac{\lambda}{C} QoS$$

(Formula 5.2)

where λ is the throughput, *C* is the cost and *QoS* represents a set of parameters that measures the quality of service perceived by the users. Therefore, it is necessary to define these three functions (λ, *QoS*, and *C*), taking into consideration a scale factor *k*.

In our proposed MAS, the scale factor k should be considered as the number of P and M agents in the network, which is identical to the number of network nodes and the number of LPs respectively. It is also necessary to evaluate the function *F* with different network sizes and different numbers of LPs. For this purpose we have defined the networks proposed in Figure 5.20. Note that the scalability factor *k* is proposed to be the length of the side of the squared network.

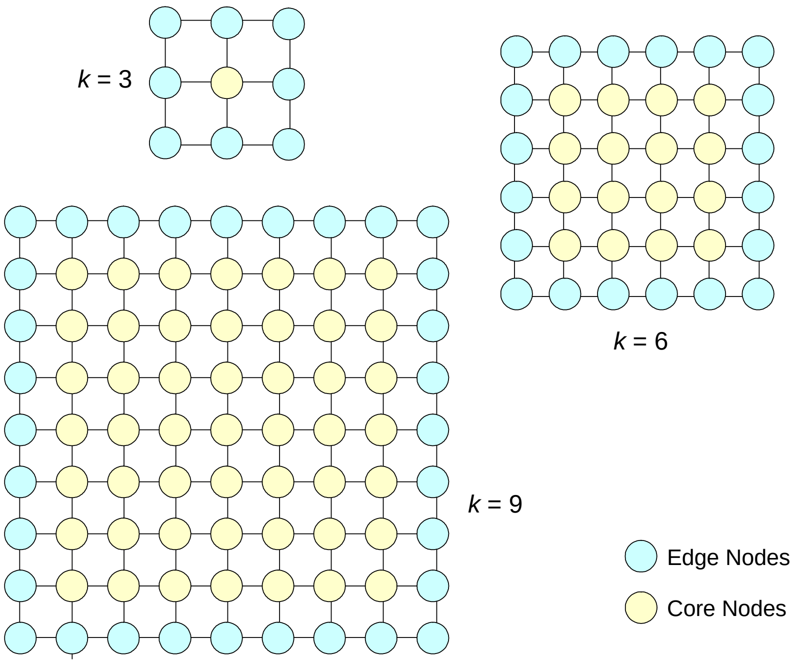


Fig. 5.20: Networks for the Scalability Evaluation.

where each network has the following properties:

- number of nodes $n = k^2$
- number of edge nodes $e = 4*k-4$
- number of unidirectional LPs $nLP = e\cdot(e-1)$

  (this is because we suppose that the edge nodes constitute a full meshed logical network)
- mean length (number of hops) of the LPs: $lLP = (4k^2-8k-4) / (e-1)$

  (the addition of all the lengths of the LPs from one edge node – the origin – to all the other edge nodes, divided by the number of edge nodes except the origin edge node)

The scalability of the MAS basically depends on the amount of processing to be done by each P agent. This mainly depends on the management traffic due to the action of the P agents performing bandwidth readjustments and LP rerouting, and the number of P agents involved. The possible effect due to network failures is considered not to be significant in this study and is omitted. We are also omitting the weight due to the number of M agents in the edge nodes, because they are lightweight processes (threads), which perform simple calculations and they are not active most of the time (this depends on the monitoring period).

Note that all the constant values used in the following definitions are derived from our experience in the implementation and simulation of the proposed architecture. In addition, it is worth saying that the behaviour of the resulting mathematical model mainly depends on the variable values.

The function *C* represents the amount of management traffic processed by each P agent, and is defined as:

$$C = \frac{\begin{bmatrix}\text{partial network} \\ \text{view size}\end{bmatrix} \cdot \begin{bmatrix}\text{number of} \\ \text{messages}\end{bmatrix}}{[\text{number of nodes}]} = \frac{M \cdot nLP \cdot \alpha \cdot \sum_{LP} (CBP_{LP} \cdot OC_{LP})}{n^2} \qquad \text{(Formula 5.3)}$$

where M represents the size (bytes) of the representation of a single LP (this depends on the mean LP length plus a fix part), and it is defined as $M = 10 + 2 \cdot lLP$. The partial network view that a P agent has is defined here as the knowledge of the P agent about the LPs that traverse the node it manages. Therefore, the factor $M \cdot nLP/n$ in Formula 5.3 is the size of the representation of a partial network view, and it is equal to the size of a management message containing a partial network view.

The remaining factors of the numerator $\alpha \cdot \sum_{LP} (CBP_{LP} \cdot OC_{LP})$ represent the number of management messages, where α represents the number of messages needed to solve one congestion situation, CBP_{LP} is the connection blocking probability for each LP, and OC_{LP} is the offered connections for each LP. This model considers that all the connection rejections provoke congestion detection and, therefore, a bandwidth readjustment.

The term α depends on the algorithms used for solving a congestion situation. It is supposed that the MAS uses the Any-LP algorithm combined with the re-routing mechanism. The possible situations are divided into three groups (probabilities p_1, p_2 and p_3). In group 1, it is supposed that the congestion situation is solved using available bandwidth from the physical links and only 2 messages are used. In group 2, it is supposed that the congestion situation is solved by using underused bandwidth from another LP. In this case, it is also considered that all the LPs going through a node are checked. Therefore the number of messages generated depends on the number of LPs. It is supposed that these LPs are uniformly distributed throughout the network and that every P agent must check *nLP/n* LPs. In the third group, it is supposed that the messages used in the re-routing procedure are proportional to the mean LP length (*lLP*). Therefore α is defined as follows:

$$\alpha = 2 \cdot p_1 + \frac{nLP}{n} \cdot p_2 + lLP \cdot p_3 \qquad \text{(Formula 5.4)}$$

The function λ represents the total amount of dispatched connections, and it is defined as:

$$\lambda = \sum_{LP} (OC_{LP} \cdot (1 - CBP_{LP})) \qquad \text{(Formula 5.5)}$$

The function *QoS* represents the time spent in solving a congestion situation compared with a target time t_0. It is similar to the function *QoS* defined in [Jogalekar and Woodside, 1998]. However, it is not realistic to expect the same response time in a small network as in a large network, because this is not an on-line service that requires the fastest response time possible in all situations, neither is it a real-time mechanism. Therefore, we considered that the target time is acceptable if the congestion situation is solved in a time limit depending on the LP length. We have defined the target time as a function of the mean LP length $t_0 = 3 \cdot lLP$. The function *QoS* is defined as:

$$QoS = \frac{1}{1 + \frac{\alpha \cdot lLP}{t_0}} = \frac{3}{3 + \alpha} \qquad \text{(Formula 5.6)}$$

Therefore the function *QoS* just depends on the difficulty (number of messages used) of solving a congestion problem. In order to simplify the model it is supposed that the CBP_{LP} and OC_{LP} are the same for all the LPs (*CBP* and *OC*). In this case, the functions *C*, λ, and *QoS* can be expressed as:

$$C = \frac{M \cdot nLP^2 \cdot \alpha \cdot CBP \cdot OC}{n^2} \qquad \text{(Formula 5.7)}$$

$$\lambda = nLP \cdot OC \cdot (1 - CBP) \qquad \text{(Formula 5.8)}$$

$$QoS = \frac{3}{3+\alpha} \qquad \text{(Formula 5.9)}$$

After the definition of the three functions, their combination using function *F* results, after simplification, in the following expression:

$$F = \frac{\lambda}{C} QoS = \frac{3n^2 \cdot (1-CBP)}{M \cdot nLP \cdot \alpha \cdot CBP \cdot (3+\alpha)} \qquad \text{(Formula 5.10)}$$

Function *F* depends on the scalability factor *k*: *F(k)*. However, as stated in [Jogalekar and Woodside, 1998] the scalability of a system should be evaluated comparing different scales of the system (Formula 3.1). Therefore, function *F* is calculated for different scale factors *k* starting from *k=10* and every one of them is compared with this initial value of *F* for *k=10*. The results are shown in Figure 5.21.

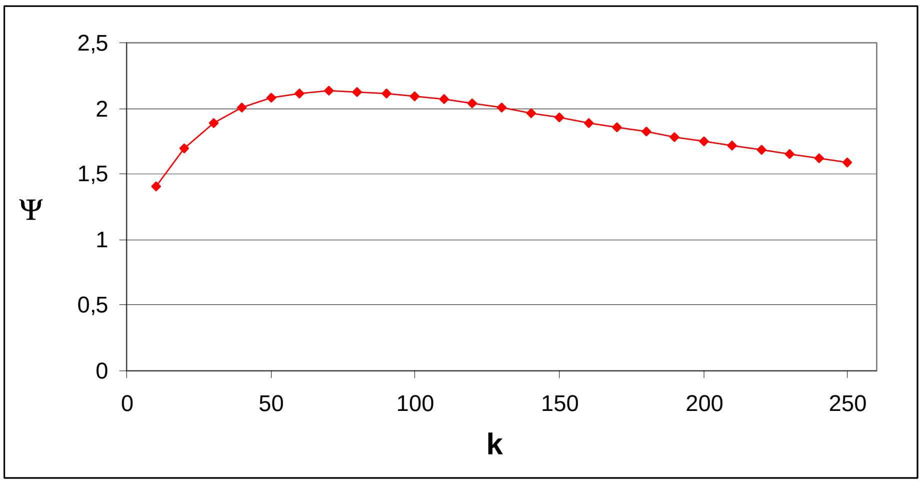


Fig. 5.21: Scalability Evaluation.

The results obtained from the proposed mathematical model show that the MAS is not scalable (i.e. when $k \to \infty$ then $\Psi \to 0$), although it shows a suitable scalability using a wide range of scale factor *k*. This is because the model has a tendency to be very slow when approaching the zero value. This means that the system operation is fine with large networks (e.g. when k=100, there are 10000 nodes and 156420 LPs in the network), however, it seems reasonable that several design improvements could help in the achievement of better scalability. For instance, to reduce the number of transmissions of the partial network view and increase the autonomy of the P agents when making their decisions. It could also be interesting to study if it is possible to define a mechanism that allows the transmission of only the updated information from the last message between two P agents, and not the whole partial network view.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

This thesis addresses the fact that every day more there is a need for automatic reconfiguration of communication networks, since manual or periodical reconfiguration cannot deal with unforeseen problems. Therefore, we have detected and handled the need for coordination between automated management mechanisms acting on the same resources. Moreover, automatic reconfiguration of the Logical Network must be carried out carefully and only when it is really needed. In this thesis a dynamic logical path management architecture for network resource management is proposed. The proposed architecture deals with management functions that make use of the logical path concept with the objective of integrating and coordinating them. The main resource to be managed for the proposed architecture is bandwidth, which is used for several management functions. The main functions addressed in this thesis are:

- **load balancing**, which is based on the minimisation of the Connection Blocking Probability for each logical path. It performs the bandwidth reallocation and the LP rerouting mechanisms
- **fault protection management** using pre-planned protection mechanisms based on global backup paths
- **spare capacity management** of the backup paths, which also enables the possibility of sharing bandwidth given a certain network protection level

The proposed architecture uses Software Agents and performs completely distributed and automated management. The architecture is organised as a Multi-Agent System (MAS) including two types of agents: the Monitoring Agents (M agents) and the Performance Agents (P agents). The MAS is physically distributed over all the network nodes and performs lightweight monitoring of the established LP in order to detect congestion (e.g. a certain level of connection rejection). The fact that the load balancing actions (bandwidth reallocation, LP rerouting) are carried out dynamically, only when necessary, means that there are not any unnecessary reconfigurations and this helps in network performance and system scalability. The load balancing mechanisms are effectively coordinated by the proposed MAS with the pre-planned restoration mechanism and the spare capacity management reserved for the backup paths.

The results presented show the ability of the architecture to carry out all these tasks in a coordinated way. Moreover, the proposed MAS achieves good scalability and robustness (inherent in distributed systems) but also introduces the idea of modularity and independence with respect to other management mechanisms. It is not the goal of this thesis to propose new mechanisms, for instance for load balancing or fault restoration, however, in order to test the proposed architecture several simple mechanisms have been introduced as examples that achieve another objective of our proposal: simplicity. Moreover, in other implementations of the proposed architecture, it is possible to use other kinds of decision-making mechanisms and, therefore, the objective of flexibility is also achieved. Regarding flexibility, the proposed architecture can be used only in some parts of the network or in some sub-networks if necessary. The proposed architecture can be switched on or off without disrupting or interfering the network operation in normal conditions (no failures).

Many experiments have been carried out in order to test all the functionalities of the proposed architecture. Several experiments have been carried out in order to verify that the implemented mechanisms were operating properly and others have been proposed in order to show the behaviour of the MAS. Many network scenarios have been used with very high offered traffic that loads the MAS heavily and provokes a large number of readjustments in the logical network. These experiments confirm the capacity of the proposed architecture to perform the desired tasks in a suitable way.

Results obtained from the monitoring and congestion detection mechanisms show that simple lightweight monitoring is enough for the detection of congestion situations in the logical paths. In this case, exhaustive simulations using different parameters revealed that the lower connection blocking ratios are achieved by the Load function, large step sizes and mid-to-low monitoring periods. However, the difficulty in tuning the thresholds suggests that the Monitoring agents could automatically adapt these monitoring parameters (this is presented as future work).

The load balancing experiments have demonstrated the ability of the proposed bandwidth reallocation algorithms to increase/decrease the bandwidths of the logical paths and adapt them to the offered load. These bandwidth reallocation algorithms combined with the proposed re-routing algorithm can manage the network resources and maintain the connection blocking ratios at lower levels.

Despite the P agents not having a global network view , which means that the optimal logical network cannot be guaranteed, the proposed mechanisms achieve their objective. Results show that the bandwidth of the logical paths can be increased/decreased taking into account that the spare bandwidth assigned to their backups is also adapted. Moreover, when a logical path is re-routed its backup path is considered in the re-routing process in order to maintain the paths disjoint. Experiments combining the monitoring and congestion detection, the load balancing, the fault protection and the spare capacity management mechanisms altogether, have been carried out showing the ability of the proposed architecture to manage these mechanisms. More specifically, the results obtained in the experiments presented in section 5.4 confirm that all the mechanisms used (load balancing, network protection, spare capacity management) show the benefits of using the proposed architecture, in other words, the success of applying all the mechanisms dynamically and automatically without interference and in a co-ordinated way.

6.2 Future Work

There are many issues that have been left as future work throughout this thesis. The most significant ones are the following:

- The work presented in this thesis proposes a completely distributed architecture which is automated and can operate without any kind of human intervention. An open issue of automated network management systems is how they can interact with the human network administrator, and how the configuration commands or indications can be downloaded to the distributed MAS and the status and conditions of the agents uploaded to the network administrator's front-end. The purpose is that the human administrators would be able to introduce orders and objectives into the system and that the system would be able to give some sort of vision of the network reconfigurations and detected problems.
- It could be interesting to extend the idea of LP congestion to the physical links. Suppose that a network node has an outgoing physical link where all, or almost all the bandwidth is already assigned between LPs and backup LPs. In this case, the P agent responsible for that physical link could consider, under several circumstances (e.g. several LPs going through that physical link have recently tried to increase their bandwidth and have failed), that the physical link is congested and it can take initiatives in order to try to alleviate that link. For instance, it could be interesting to initiate the rerouting of the backup LPs (which in our implementation were fixed) and

/or send warning messages to the origin nodes of the affected LPs informing them of the situation. The P agents at the origin nodes can then make their own decisions if the affected LPs are congested or near congestion.

- In the current implementation selecting the Triggering Function, the monitoring period, the limit of the function, and the step size, has been left static and manually fixed by the human network administrator. However, it would be interesting that the M agents could make these decisions on their own and that these parameters could become dynamic. This could depend on the behaviour of the offered load, the class of offered load, the goals given to the M agents, the monitored LP priority, the available bandwidth in the outgoing link, even the node processing load, etc. This could imply the use of heuristics or even learning mechanisms in the M agents.
- The possible interaction of a system based on the proposed architecture that manages the resources of a network or sub-network with other systems of the same type managing neighbour networks or sub-networks, e.g. belonging to different network providers. This implies the use of standard agent communication protocols and the development of a network management ontology. We have already started to work in this line [Cots et al., 2003] with a project that involves the management of backup LPs that go through different network domains belonging to different network providers.
- A further study of the interactions with other management and control mechanisms acting in the same network. The interaction with the Connection Admission Control is particularly interesting. It could be useful to distinguish the class of traffic of the rejected calls in order to avoid, as far as possible, any kind of unfairness between them. That is, instead of applying the TF to all the connections accepted/rejected by an LP, apply it distinguishing between the classes of traffic. The interactions with the mechanisms used for establishing and releasing LPs could also be studied.
- The desired objective of achieving good scalability suggested the use of completely distributed management architecture without a global network view. The lack of a global network view means that an optimal logical network configuration cannot be guaranteed. It could be interesting to study if other distributed problem solving mechanisms can be adapted and applied to this specific scenario in order to help achieve logical network reconfigurations that is nearer to the optimal one.

References

[Albayrak and Garijo, 1998] Albayrak S., Garijo F.J. (Eds.), “Intelligent Agents for Telecommunication Applications”, Proceedings of Second International Workshop, IATA'98, Paris, France, July 4-7, 1998, Springer-Verlag, ISBN: 3-540-64720-1, Lecture Notes of Artificial Intelligence vol. 1437.

[Albayrak, 1998] Albayrak S. (Ed.), “Intelligent Agents for Telecommunications Applications”, IOS Press 1998, ISBN 90 5199 295 5.

[Albayrak, 1999] Albayrak S. (Ed.), “Intelligent Agents for Telecommunication Applications” Proceedings of Third International Workshop, IATA'99, Stockholm, Sweden, August 9-10, 1999, Springer-Verlag, ISBN 3-540-66539-0, Lecture Notes of Artificial Intelligence vol. 1699.

[Anderson et al., 2001] “LDP Specification”, L. Andersson, P. Doolan, N. Feldman, A. Fredette, B. Thomas, RFC 3036, January 2001.

[Aneroussis, 1996] Nikolaos Anerousis, “Managing Virtual Circuit and Virtual Path Services on ATM Networks with Quality of Service Guarantees”, PhD Thesis. Graduate School of Arts and Sciences, Columbia University, 1996.

[Ashwood-Smith and Berger, 2003] P. Ashwood-Smith, L. Berger, “Generalized Multi-Protocol Label Switching (GMPLS) Signalling Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions”, RFC 3472, 2003.

[ATM Forum, 1996] ATM Forum af-tm-0056.000, “Traffic Management Specification version 4.0”, 1996. URL: http://www.atmforum.com.

[Awduche et al., 1999] D. Awduche, J. Malcolm, J. Gogbua, M. O’Dell and J. McManus “Requirements for Traffic Engineering Over MPLS”,. RFC 2702, September 1999.

[Awduche et al., 2002] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja and X. Xiao, “Overview and Principles of Internet Traffic Engineering”, RFC 3272, May 2002.

[Banerjee et al., 2001] Banerjee A., Drake J., Lang J.P., Turner B. Kompella K., Rekhter Y., “Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements”, IEEE Communications Magazine, January 2001.

[Berger et al., 2003a] L. Berger (Ed), “Generalized Multi-Protocol Label Switching (GMPLS) Signalling Functional Description”, RFC 3471, 2003.

[Berger et al., 2003b] L. Berger (Ed), “Generalized Multi-Protocol Label Switching (GMPLS) Signalling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions”, RFC 3473, 2003.

[Bieszczad et al., 1998] A. Bieszczad, B. Pagurek, T. White, “Mobile Agents for Network Management”, IEEE Communications Surveys, vol. 27, 4Q, 1998 http://www.comsoc.org/pubs/surveys.

[Bigham et al., 1999] Bigham J., Cuthbert L.G., Hayzelden A.L.G., Luo Z., “Multi-Agent System for Network Resource Management”, International Conference on Intelligence in Services and Networks, IS&N'99, Barcelona (Spain), April 1999.

[Black, 1994] U. Black, “Network Management Standards”, 2nd ed., McGraw Hill, 1994, ISBN 0-07-005570-X

[Black, 1995] Uyless Black, “ATM: Foundation for Broadband Networks”, Prentice Hall, 1995. ISBN: 0-13-297178-X.

[Blake et al., 1998] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An Architecture for Differentiated Services”, RFC 2475, December 1998.

[Bodanese and Cuthbert, 1999] Bodanese E.L., Cuthbert L.G., "Distributed channel allocation scheme for cellular networks using intelligent agents", 7th International Conference in Telecommunications Systems, 18-21 March 1999, Nashville, Tennessee (USA).

[Bohoris et al., 2000] C. Bohoris, G. Pavlou, H. Cruickshank, "Using mobile agents for network performance management", Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP , 10-14 April 2000.

[Bonabeau et al., 1999] Eric Bonabeau, Marco Dorigo, Guy Theraulaz, "Swarm Intelligence: From Natural to Artificial Systems", Oxford University Press, 1999, ISBN: 0-19-513159-2.

[Braden et al., 1997] "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification ", R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, RFC 2750, September 1997.

[Burness et al., 1999] A-L. Burness, R. Titmuss, C. Lebre, K. Brown, A. Brookland, "Scalability evaluation of a distributed agent system", Distributed Systems Engineering 6, The British Computer Society, IEE and IOP Publishing Ltd., 1999.

[Calle et al., 2001] E. Calle, T. Jové, P. Vilà, J.L. Marzo, "A Dynamic Multilevel MPLS Protection Domain", 3rd International Workshop on Design of Reliable Communication Networks, DRCN, Budapest (Hungary), 2001.

[Caripe et al., 1998] Wilmer Caripe, George Cybenko, Katsuhiro Moizumi, Robert Gray, "Network Awareness and Mobile Agent Systems", IEEE Communications Magazine, July 1998.

[CCITT Rec. I.321, 1991] CCITT Recommendation I.321, "B-ISDN Protocol Reference Model and its Application", Geneva 1991.

[CCITT Rec. M.3010, 1992] "Recommendation M.3010: Principles for a Telecommunications Management Network", Comité Consultative Internationale de Telegraphique et Telephonique, Geneva 1992.

[Cheikhrouhou et al. 1998] M. Cheikhrouhou, P. Conti, J. Labetoulle, "Intelligent Agents in Network Management, A State of the Art". Networking and Information Systems, Vol. 1,N 1, 1998.

[Cheikhrouhou, 1999] M. Cheikhrouhou. "A flexible agent architecture applied to Network Management", HPOVUA'99, Bologna, Italy. June 13-15, 1999.

[Chen and Liu, 1994] Thomas M. Chen, Steve S. Liu, "Management and Control Functions in ATM Switching Systems", IEEE Network, vol 8 no 4, July/August 1994.

[Chen and Oh, 1999] T. Chen, T. Oh, "Reliable Services in MPLS", IEEE Communications Magazine, Vol. 37, No. 12, December 1999.

[Chen et al., 1996] Thomas M. Chen, Steve S. Liu, David Wang, Vijay K. Samalam, Michael J. Procanik, Dinyar Kavouspour, "Monitoring and Control of ATM Networks Using Special Cells", IEEE Network, vol 10 no 5, September/October 1996.

[CORBA URL] Common Request Broker Architecture: http://www.corba.org.

[Corley et al. 2000] S. Corley, F. Garijo, J. Hickie, "Agent-Based Operational Support Systems", In proceedings of International Conference on Intelligence in next generation Networks, ICIN'2000, 2000.

[Cots et al., 2003] Santiago Cots, Teodor Jové, Pere Vilà, "A Call-level Network Simulator Framework based on a Standard Agent Platform", In Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2003), Montreal (Canada), July 20-24, 2003

[Davie and Rekhter, 2000] B. Davie and Y. Rekhter, "MPLS Technology and Applications", Academic Press – Morgan Kaufmann Publishers, 2000. ISBN: 1-55860-656-4.

[Davison et al., 1998] Davison R.G., Hardwicke J.J., Cox M.D.J., "Applying the agent paradigm to network management", BT Technology Journal, vol 16 no 3, July 1998.

[Di Caro and Dorigo 1997] G. Di Caro and M. Dorigo, "AntNet: a mobile agents approach to adaptive routing", Tech. Rep. IRIDIA/97-12, Universite Libre de Bruxelles, Belgium

[DIANA URL] DIANA project URL: http://www.eurecom.fr/~diana/

[Du et al., 2003] Timon C. Du, Eldon Y. Li, An-Pin Chang, "Mobile Agents in Distributed Network Management", Communications of the ACM, Vol.46 No.7, July 2003.

[Dziong et al., 1996a] Z. Dziong, Y. Xiong, L.G. Mason, "Virtual Network Concept and its applications for resource management in ATM based networks", International IFIP/IEEE Conference on Broadband Communications, Chapman & Hall, 1996.

[Dziong et al., 1996b] Z. Dziong, J. Zhang, L.G. Mason, "Virtual Network Design – An Economic Approach", Proceedings of 10th ITC Specialist's Seminar on "Control in Communications", Sweden, 1996.

[Dziong, 1997], Z. Dziong, "ATM Network Resource Management", McGraw Hill 1997, ISBN: 0-07-018546-8

[Eurescom P712 URL] Eurescom P712 URL: http://www.eurescom.de/public/projects/P700-series/P712/P712.htm.

[FIPA URL] Foundation for Intelligent Physical Agents, URL: http://www.fipa.org.

[Friesen et al., 1996] Friesen V.J., Harms J.J., Wong J.W., "Resource Management with Virtual Paths in ATM networks", IEEE Network, vol 10 no 5, September/October 1996.

[Gavalas et al., 2000] D. Gavalas, D. Greenwood, M. Ghanbari, M. O'Mahony, "Advanced Network Monitoring Applications Based on Mobile/Intelligent Agent Technology", Computer Communications 23, pp.720-730, 2000.

[Gibney et al., 1999] M. A. Gibney, N. R. Jennings, N. J. Vriend, J. M. Griffiths, "Market-based call routing in telecommunications networks using adaptive pricing and real bidding" in Proceedings of the 1999 Conference on Intelligent Agents for Telecommunications Applications, Stockholm, Sweden, August 1999.

[Goldszmidt and Yemini, 1998] G. Goldszmidt, Y. Yemini "Delegated Agents for Network Management", IEEE Communications Magazine, March 1998.

[Greenberg and Byington, 1998] Michael S. Greenberg and Jennifer C. Byington. Mobile Agents and Security. IEEE Communications Magazine, 36(7):76--85, July 1998.

[Guerin et al., 1991] R. Guerin, H. Ahmadi, M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high-speed networks", IEEE Journal on Selected Areas in Communications, Vol. 9, No. 7, September 1991, pp. 968-981.

[Guerin et al., 1997] R. Guerin, D. Williams, A. Orda "QoS Routing Mechanisms and OSPF Extensions", Proceedings of GLOBECOM. November 1997.

[Halls and Rooney, 1998] David A. Halls, Sean G. Rooney, "Controlling the Tempest: Adaptive Management in Advanced ATM Control Architectures", IEEE Journal on Selected Areas in Communications, April 1998

[Hardwicke and Davison, 1998] Hardwicke J., Davison R., "Software Agents for ATM Performance Management", IEEE NOMS'98 Network Opertaions and Management Symposium, New Orleans (USA), February 1998.

[Hayzelden and Bigham, 1998] Hayzelden A.L.G., Bigham J., "Heterogeneous Multi-Agent Architecture for ATM Virtual Path Network Resource Configuration", Intelligent Agents for Telecommunications Applications (IATA 98), June 1998.

[Hayzelden and Bigham, 1999a] Hayzelden A., Bigham J. (Eds.), "Software Agents for Future Communications Systems", Springer-Verlag 1999, ISBN 3-540-65578-6.

[Hayzelden and Bigham, 1999b] A.L.G. Hayzelden, J. Bigham, "Agent Technology in Communications Systems: An Overview", Knowledge Engineering Review, vol. 14.4, 1999.

[Hayzelden and Bourne, 2001] Hayzelden A.L.G., Bourne R.A. (Eds.), "Agent Technology for Communications Infrastructure", 2001, John Wiley & Sons Ltd, ISBN 0-471-49815-7.

[Hayzelden, 1998] Hayzelden A.L.G., "Telecommunications Multi-Agent Control System (Tele-MACS)", European Conference on Artificial Intelligence (ECAI 98), August 1998.

[IETF URL] http://www.ietf.org

[ISO 10040, 1992] "Information Processing Systems – OSI Systems Management Overview", Geneva, 1992.

[ISO 7498-4, 1989] "Information Processing Systems – OSI Basic Reference Model Part 4: Management Framework", Geneva, 1989.

[ISO 9596, 1991] “Information Processing Systems – OSI Common Management Information Protocol”, Geneva 1991.

[ITU-T Rec. I.371, 1995] ITU-T Recommendation I.371, “Traffic Control and Congestion Control in B-ISDN”, Geneva, 1995.

[Jamoussi et al., 2002] "Constraint-Based LSP Setup using LDP", B. Jamoussi, L. Andersson, R. Collon and R. Dantu, RFC 3212, January 2002.

[Jardin, 1996] P. Jardin, “Supporting Scalability and Flexibility in a Distributed Management Platform”, Distributed Systems Engineering 3, The British Computer Society, IEE and IOP Publishing Ltd., 1996.

[Java URL] Sun Java, http://java.sun.com/

[JDMK URL] Java Dynamic Management Kit: http://java.sun.com/products/jdmk.

[Jennings, 2001] N.R. Jennings, “An Agent-Based Approach for building complex software systems”, Communications of the ACM, Vol.44 No.4, pp.35-41, 2001.

[Jogalekar and Woodside, 1998] P. Jogalekar and M. Woodside, “Evaluating the scalability of Distributed Systems”, Proc. 31st Annual Hawaii International Conference on System Sciences, IEEE Computer Society Press, vol. 7 (January 1998), pp. 524-531.

[Jogalekar and Woodside, 2000] P. Jogalekar and M. Woodside, “Evaluating the scalability of Distributed Systems”, IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 6 (June 2000), pp. 589-603.

[Kahani and Beadle, 1997] M. Kahani, H. Beadle, “Decentralised Approaches for Network Management”, Computer Communications Rev., vol. 27 pp. 36-47, July 1997.

[Kawamura and Ohta, 1999] R. Kawamura, H. Ohta, “Architectures for ATM Network Survivability and Their Field Deployment”, IEEE Communications Magazine, August 1999.

[Kim et al., 1994] J.B. Kim, T. Suda, M. Yoshimura, “International Standardization of B-ISDN”, Computer Networks and ISDN Systems, 27, 1994.

[Kyas, 1995] Othmar Kyas, “ATM networks”, International Thomson Computer Press, 1995, ISBN 1-85032-128-0.

[Lang, 2003] “Link Management Protocol (LMP)” J. Lang (Ed), Work in Progress - Internet Draft, June 2003.

[Larsson and Arvidsson, 1999] Sven-Olof Larsson, Ake Arvidsson, “An Adaptive Local Method for VPC Capacity Management”, ITC 16, Key and Smith (Eds.), Elsevier Science B.V., 1999.

[Le Boudec, 1992] Jean-Yves Le Boudec, “The Asynchronous Transfer Mode: a tutorial”, Computer Networks and ISDN Systems, vol 24, no 4, May 1992.

[Lee et al. 1998] L.C. Lee, H.S. Nwana, D.T. Ndumu, P. De Wilde, “The stability, scalability and performance of multiagent systems”, BT Technology Journal, vol. 16 no. 3 (July 1998).

[Leon-Garcia and Mason, 2003] A. Leon-Garcia, L.G. Mason, “Virtual Network Resource Management for Next-Generation Networks”, IEEE Communications Magazine, Vol.41, No.7, July 2003.

[Liu et al. 2001] B. Liu, D.R. Figueiredo, Y. Guo, J. Kurose, D. Towsley, “A Study of Networks Simulation Efficiency: Fluid Simulation vs. Packet-Level Simulation”, Proceedings of INFOCOM, Anchorage, Alaska, USA, April 2001.

[Luck et al., 2003] Michael Luck, Peter McBurney, Chris Preist, “Agent Technology: Enabling Next Generation Computing”, AgentLink II European Network of Excellence for Agent-based Computing (IST-1999-29003), Technical Document, 2003. ISBN 0854-327886.

[Luo et al., 1999] Luo Z., Bigham J., Cuthbert L.G., Hayzelden A.L.G., “Traffic Control and Resource Management using a Multi-Agent System”, 5th International Conference on Broadband Communications, Hong Kong (China), November 1999.

[Magedanz and Eckardt, 1996] T. Magedanz, T. Eckardt, “Mobile Software Agents: A new Paradigm for Telecommunications Management”, Network Operations and Management Symposium (NOMS), 1996.

[Magedanz et al., 1996] T. Magedanz, K. Rothermel, S. Krause, “Intelligent Agents: An Emerging Technology for Next Generation Telecommunications?”, INFOCOM’96, San Francisco, USA, March 1996.

[Mannie, 2003] “Generalized Multi-Protocol Label Switching Architecture”, Eric Mannie (Ed), Work in Progress - Internet Draft (http://www.ietf.org), May 2003.

[Martin-Flatin and Znaty., 2000] J. Martin-Flatin, S. Znaty, “Two Taxonomies of Distributed Network and Systems Management Paradigms”, Emerging Trends and Challenges in Network Management, Plenum Press, New York, Chapter 3, March 2000.

[Marzo et al. 2003b] J.L. Marzo, E. Calle, C. Scoglio, T. Anjali, "QoS On-Line Routing and MPLS Multilevel Protection: a Survey", IEEE Communication Magazine, Vol.41 No.10, pp. 126-132, October 2003.

[Marzo et al., 2003a] J.L. Marzo, P. Vilà, L. Fàbrega, D. Massaguer, “A Distributed Simulator for Network Resource Management Investigation”, Computer Communications Journal (Elsevier) Vol.26, No.15, pp. 1782-1791, September 2003.

[Marzo, 1996] J.L. Marzo, “Enhanced Convolution Approach for CAC in ATM Networks, an Analytical Study and Implementation”, PhD Thesis, University of Girona, 1996.

[MIMIC URL] Gambit Communications MIMIC, http://www.gambitcomm.com/

[Moy, 1998] “OSPF Version2”, J. Moy, RFC 2328, April 1998.

[Muller, 1997] Nathan J. Muller, “Improving Network Operations with Intelligent Agents”, International Journal of Network Management, Vol.7, pp.116-126, 1997.

[Nwana, 1996] H.S. Nwana, “Software agents: an overview”, Knowledge Engineering Review Vol.11 No.3, pp.205-244, 1996.

[Oh et al., 2000] T.H. Oh, T.M. Chen, J.L. Kennington, “Fault Restoration and Spare Capacity Allocation with QoS Constraints for MPLS Networks”, GLOBECOM 2000.

[Oliveira et al., 1996] Raul Oliveira, Dominique Sidou and Jacques Labetoulle. “Customized network management based on applications requirements”. In Proceedings of the First IEEE International Workshop on Enterprise Networking - ENW '96, Dallas, Texas, USA, June 27 1996.

[Oran, 1990] “OSI IS-IS Intra-domain Routing Protocol”, D. Oran, Editor, RFC 1192, February 1990.

[OSF, 1992] “The OSF Distributed Management Environment Architecture”, Open Software Foundation, May 1992.

[Pras, 1995] Aiko Pras PhD Thesis “Network Management Architectures”, University of Twente, The Netherlands 1995. ISBN 90-365-0728-6.

[Press et al., 1992] Press W.H., Flannery B.P., Teukolsky S.A. and Vetterling W.T., “Numerical Recipes in C” 2nd Ed., Cambridge University Press 1992. ISBN: 0-521-43108-5. Book available on-line at http://www.library.cornell.edu/nr/bookcpdf.html

[Qureshi and Harms, 1997] Rashid Qureshi, Janelle Harms, “A Flexible Virtual Path Topology Design Algorithm”, IEEE GOBECOM'97 Global Telecommunications Conference – Phoenix (USA), November 1997

[Raman, 1998] L. Raman, “OSI Systems and Network Management”, IEEE Communications Magazine, March 1998.

[Rekhter and Li, 1995] “A Border gateway Protocol (BGP-4)”, Y. Rekhter and T. Li, RFC 1771, March 1995.

[RFC 1157, 1990] J.D. Case, M. Fedor, M.L. Schoffstall “Simple Network Management Protocol”, May 1990.

[Rosen et al., 2001] “Multiprotocol Label Switching Architecture”, E. Rosen, A. Viswanathan, R. Callon. RFC 3031, January 2001.

[Rosen et al., 2001b] “MPLS Label Stack Encoding”, E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, A. Conta. RFC 3032, January 2001.

[Sahai and Morin 1999] A. Sahai and C. Morin, “Mobile Agents for Managing Networks: The MAGENTA perspective”, in Future Agents for Network Communication Systems, pp 358-389, Springer, 1999.

[Sato et al., 1990] Ken-ichi Sato, Satoru Ohta, Ikuo Tokizawa, “Broad-Band ATM Network Architecture Based on Virtual Paths”, IEEE Transactions on Communications, vol 38 no 8, August 1990

[Sato et al., 1991] Naoshi Sato, Tsukasa Okamoto, Tadahiro Yokoi, “Performance Monitoring Method using OAM Cells in B-ISDN”, IEEE Supercomm ICC'91 , June 1991.

[Schoonderwoerd et al. 1996] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, “Ant-based load balancing in telecommunications networks”, HP-Labs Technical Report HPL-96-76, May 1996 (http://www.hpl.hp.com/techreports/96/HPL-96-76.html)

[Sharma et al. 2002] V. Sharma, B. Crane, S. Makarn, K. Owens, C. Huang, F. Hellstrand, J. Weil, L. Anderson, B. Jarnoussi, B. Cain, S. Civanlar and A. Chiu, “Framework for MPLS-based recovery”, Internet Draft, July 2002.

[Sidor, 1998] David J. Sidor, “TMN Standards: Satisfying Today's Needs While Preparing for Tomorrow”, IEEE Communications Magazine, March 1998.

[Siegl and Trausmuth, 1996] M. Siegl, G. Trausmuth, “Hierarchical Network Management: a Concept and its Prototype in SNMPv2”, Computer Networks & ISDN Systems, vol. 28 pp. 441-452, April 1996.

[Silva et al 1999] L. M. Silva, P. Simões, G. Soares, P. Martins, V. Batista, C. Renato, L. Almeida, N. Stohr, “JAMES: A Platform of Mobile Agents for the Management of Telecommunication Networks“, in Proceedings of the Third International Workshop on Intelligent Agents for Telecommunication (IATA'99), August 1999.

[Sloman, 1996] Morris Sloman (Ed.) “Network and Distributed Systems Management”, Addison-Wesley 1996. ISBN 0-201-62745-0.

[Sohn and Sung, 2002] Kyu Seek Sohn, Dan Keun Sung, “A Distributed LPS Mechanism to Reduce Spare Bandwidth in MPLS Networks”, ICC 2002.

[Somers et al., 1997] Somers F., Evans R., Kerr D., O’Sullivan D., “Scalable low-latency network management using intelligent agents”, ISS’97, XVI World Telecom Congress, September 1997.

[Somers, 1996] Fergal Somers, “HYBRID: Unifying Centralised and Distributed Network Management using Intelligent Agents”, Network Operations and Management Symposium, April 1996.

[Stallings, 1998a] W. Stallings, “SNMP and SNMPv2: The Infrastructure for Network Management.”, IEEE Communications Magazine, March 1998.

[Stallings, 1998b] W. Stallings, “SNMP, SNMPv2, SNMPv3 and RMON 1 and 2”, Third Edition, Addison-Wesley, 1998.

[Stallings, 1998c] W. Stallings, “SNMPv3: A Security Enhancement for SNMP”, IEEE Communications Surveys • http://www.comsoc.org/pubs/surveys • Fourth Quarter 1998 • Vol. 1 No. 1

[Susilo et al., 1998] G. Susilo, A. Bieszczad, B. Pagurek, “Infrastructure for advanced network management based on Mobile Code”, Network Management and Operations (NOMS), 1998.

[Sykas et al., 1991] E.D. Sykas, K.M. Vlakos, M.J. Hillyard, “Overview of ATM networks: functions and procedures”, Computer Communications, vol 14, no 10, December 1991

[Vilà, 2001] P. Vilà, “Gestió dinàmica de recursos en xarxes de telecomunicacions utilitzant sistemes multi-agent” [in catalan] In the ACIA Newsletter, No.24 ISSN: 1577-1989, pages 7 to 32, September 2001.

[Weihmayer and Velthuijsen, 1998] R. Weihmayer, H. Velthuijsen, “Intelligent Agents in Telecommunications”, chapter 11 in N. Jennings and M. Wooldridge (Eds) “Agent Technology: Foundations, Applications and Markets”, Springer-Verlag, 1998, ISBN: 3-540-63591-2.

[Weiss, 1999] Gerhard Weiss (Ed), “Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence”, The MIT Press, 1999, ISBN: 0-262-23203-0.

[White et al., 1997] B. White B., B. Pagurek and F. Oppacher, “Connection Management by Ants: An Application of Mobile Agents in Network Management”, International Conference on Evolutionary Computation, 1997.

[White et al., 1998a] T. White, B. Pagurek, A. Bieszczad, G. Sugar, X. Tran, “Intelligent Network Modeling using Mobile Agents”, IEEE GOBECOM'98 Global Telecommunications Conference, Sydney, November 1998.

[White et al., 1998b] T. White, A. Bieszczad and B. Pagurek, "Distributed Fault Location in Networks Using Mobile Agents", Proceedings of the 3rd International Workshop on Agents in Telecommunications Applications IATA'98, Paris, France, 1998.

[White, 1997] James E. White, “Mobile Agents”, chapter 19 in “Software Agents” edited by Jeffrey M. Bradshaw, AAAI Press / MIT Press, 1997, ISBN 0-262-52234-9.

[Willmott and Faltings, 2000] Steven Willmott, Boi Faltings, “The Benefits of Environment Adaptive Organisations for Agent Coordination and Network Routing Problems”, International conference on Multi-Agent Systems, pp.333-340, 2000.

[Wooldridge and Jennings 1995] Michael Wooldridge, Nicholas R. Jennings, “Intelligent Agents: Theory and Practice”, Knowledge Engineering Review, Vol.10 No.2, pp.115-152, 1995.

[Xiao, 2000] Xipeng Xiao, “Providing Quality of Service in the Internet”, PhD Thesis, Michigan State University, 2000.

[Xiao et al., 2000] X. Xiao, A. Hannan, B. Bailey, L.M. Ni, “Traffic Engineering with MPLS in the Internet”, IEEE Network Magazine, March 2000.

[Xiong and Mason, 1999] Xiong Y., Mason L.G., “Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks”, IEEE/ACM Transactions on networking vol.7 no.1, February 1999.

[Yahara and Kawamura, 1997] T. Yahara, R. Kawamura, “Virtual Path self-healing scheme based on multi-reliability ATM network concept”, IEEE GLOBECOM’97, November 1997.

[Yahia and Robach, 1997] S. Ben Yahia, C. Robach, “Self-Healing Mechanisms in ATM Networks: The Role of Virtual Path Management Functions”, IEEE ICC'97 International Conference in Communications, June 1997.

[Yemini et al., 1991] Y. Yemini, G. Goldszmidt, S. Yemini, “Network Management by Delegation”, 2nd International Symposium on Integrated Network Management, Washington DC, April 1991.

[Yemini, 1993] Y. Yemini, “The OSI Network Management Model”, IEEE Communications Magazine vol. 31 no. 5, pages 20-29, 1993.

[Zapf et al. 1998] M. Zapf, H. Müller and K. Geihs, “Security Requirements for Mobile Agents in Electronic Markets”, Lecture Notes in Computer Science, vol 1402, 1998.

[Zhu et al., 2001] Y. Zhu, T. Chen, S. Liu, “Models and Analysis of Trade-offs in Distributed Network Management Approaches”, International Symposium on Integrated Network Management, IM 2001, Seattle (USA).

Appendix A Simulation Details

Introduction

This section details the simulation tool [Marzo et al., 2003a] developed for performing the experiments to test the proposed architecture. This simulation platform is also distributed and completely oriented to facilitate the development of network management applications both centralised and distributed. In this sense it is similar to tools like MIMIC [MIMIC URL], which is based on the simulation of SNMP Agents and is also oriented to the development of network management applications. This tool was not used for several reasons, for instance: (i) because it forces management applications to be developed based on the use of SNMP, which was not a priority in our case; (ii) due to economic reasons.

Simulations are performed at a connection level (no packet granularity). However, it is possible to generate occupation percentages for the established connections, thus giving an idea of real occupation. This is similar to the fluid simulation compared to the packet level [Liu et al. 2001], where network traffic is modelled in terms of continuous fluid flow, rather than discrete packet instances. This represents a higher level of abstraction.

The simulator can be used for testing automated management mechanisms and also manual mechanisms, demonstrating its versatility and scalability while maintaining its simplicity and ease of use. Despite it having been designed for network resource management experiments new functions can be easily added in future versions. For instance, a graphical user interface is currently under development and it is also planned to add support for the SNMP protocol.

Simulator Architecture

The distributed simulator design is based on a client/server model with many clients and servers running concurrently. The server process is called Node Emulator (NE) and its main function is the emulation of a single network node. NE processes offer high functionality to its clients which can be of different types:

- Traffic Event Generator (TEG) processes, which act as pools of users requesting connections to a destination node.

- Network Management Processes, which act monitoring and / or configuring the nodes simulated by NE processes.
- Graphical User Interfaces, which can be used to visualise, debug, etc. the data / state of the NEs
- Other

The set of TEGs and NEs constitute the distributed simulator. Other types of client processes (such as management processes) are not part of the platform (Figure 1) and can be developed using the desired programming language or platform (allowing the use of Java RMI [Java URL], CORBA [CORBA URL], etc). The access to the NE services is carried out through their API, which offers a set of functions to perform the node management, and allows the node internal status to be checked and modified.

Performing a network simulation with a certain number of nodes implies the use of the same number of NE server processes (Figure 1), which can be executed in the same or in different computers. These processes make use of the TCP/IP socket interface. Both the NE and TEG processes are implemented in C++ and can be executed in a Sun workstation under Sun Solaris OS, as well as in a PC under Linux OS.

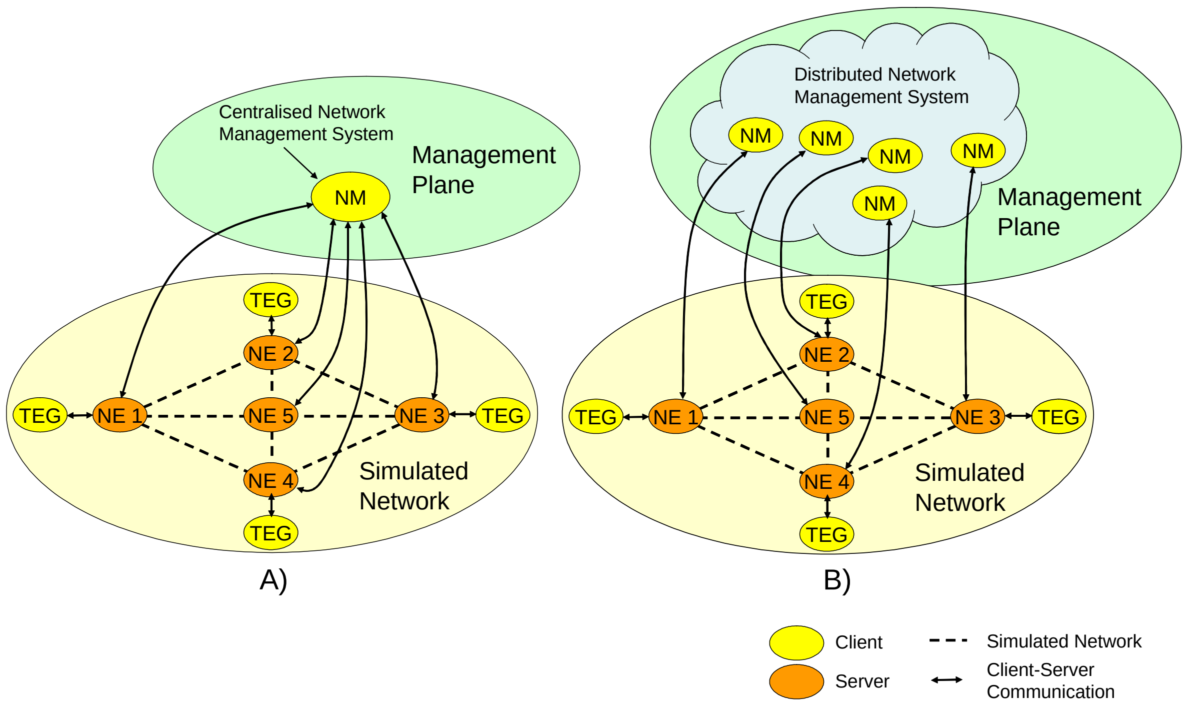


Fig. 1: Centralised (A) and Distributed (B) Management of a Simulated Network.

Every NE process generates a log file with all the functions and parameters that have been requested by clients. From the analysis of this set of log files it is possible to obtain the results and even to reproduce the complete simulation. These log files are based on formatted text and they can also be directly loaded into spreadsheet programs or statistic software.

Broadly speaking, there is no communication between NE processes unless a specific routing protocol is implemented. The default routing protocol only takes into account the direct path. However, the default routing protocol and admission control mechanism can be changed through the modification of the NE source code. These changes are relatively easy to do due to the modular object oriented simulator design.

The TEG processes are able to send events to the NE processes. The defined events include:

- connection-demand / connection-release: to establish / release user connections
- instantaneous-bandwidth: to set the percentage of the bandwidth utilised by an established connection
- link-failed / link restored: to inform the node of a failed or restored physical link, and allow the simulation of failures.

The generation of these events follows a parameterised distribution (negative exponential distribution or uniform). The different TEG processes are executed independently and all of them can be configured with different parameters. Every TEG has a pseudorandom number generator based on L'Ecuyer with Bays-Durham shuffle and added safeguards. It has a long period ($> 2.3 * 10^{18}$) and has passed all statistical presently known tests, but it is quite slow [Press et al., 1992].

As a future work on the distributed simulator itself there are three defined lines to follow. First of all we plan to develop a visual tool to make the network topology definition easier and automate the generation of the configuration files and the distribution of the simulation processes over several machines and the start of a simulation. Second, we plan the development of an automatic tool to process the log files generated for every node. This tool could merge the log files and generate statistics and extract complex results. Finally these log files could also be used to perform a debugging task in network management mechanisms and even debug the simulator itself. This last point is a long-term idea but we think it could be interesting to transform a simulation tool into a debugging tool for network management algorithms.

Appendix B   Publications and Projects

Related Publications

Journals and Book Chapters

Josep L. Marzo, **Pere Vilà** , Lluís Fàbrega, Daniel Massaguer, “A Distributed Simulator for Network Resource Management Investigation”, In **Computer Communications Journal** - Special issue on Recent Advances in Communications Networking, Volume 26, Issue 15 , September 2003, Pages 1782-1791

Pere Vilà, “Gestió dinàmica de recursos en xarxes de telecomunicacions utilitzant sistemes multi-agent” [in catalan]. In the **ACIA Newsletter** (www.acia.org), no.24 ISSN: 1577-1989, pages 7-32, September 2001

P.Vilà, J.L.Marzo, R.Fabregat, D.Harle “A Multi-Agent Approach to Dynamic Virtual Path Management in ATM Networks”. Chapter in “Agent Technology for Communications Infrastructure”, Edited by Alex L.G. Hayzelden and Rachel A. Bourne, John Wiley & Sons 2001, ISBN 0-471-49815-7, pages 167-184.

International Conferences

Santiago Cots, Teodor Jové, **Pere Vilà**, “A Call-level Network Simulator Framework based on a Standard Agent Platform”, In Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (**SPECTS 2003**), Montreal (Canada), July 20-24, 2003. Edited by Mohammad S. Obaidat, et al. SCS ISBN 1-56555-269-5. pages 218-225.

Pere Vilà, José L. Marzo, Eusebi Calle, “Dynamic Bandwidth Management as part of an Integrated Network Management System based on Distributed Agents”, In Proceedings of IEEE Global Communications Conference (**GLOBECOM 2002**), Taipei (Taiwan), November 17-21, 2002. ISBN 0-7803-7633-1.

Pere Vilà, Josep L. Marzo, Lluís Fàbrega, “Using a Multi-Agent System for Network Management”, In Proceedings of Congrés Català d’Intel•ligència Artificial (**CCIA 2002**). October 24-25, 2002. Castelló de la Plana, Spain. Butlletí de l'ACIA (ACIA Newsletter), no 28, pages 352-358, ISSN: 1577-1989, October 2002.

Pere Vilà, Josep L. Marzo, Antonio Bueno, “Automated Network Management Using a Hybrid Multi-Agent System”, In proceedings of Artificial Intelligence and Applications (**AIA 2002**), September 9-12, 2002. Málaga, Spain Edited by M.H. Hamza, ISBN: 0-88986-352-0, ISSN: 1482-7913 ACTA Press (IASTED). Pages 441-446.

Pere Vilà, Josep L. Marzo, Antonio Bueno, Daniel Massaguer, “Network Resource Management Investigation using a Distributed Simulator”, In proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (**SPECTS 2002**), San Diego, California (USA), July 14 - 18, 2002. Edited by Mohammad S. Obaidat, Franco Davoli, Ibrahim Onyuksel, and Raffaele Bolla. SCS ISBN 1-56555-252-0. pages 668-675.

Josep L. Marzo, **Pere Vilà**, Lluís Fàbrega, Daniel Massaguer, "An ATM Distributed Simulator for Network Management Research", In proceedings of 34th Annual Simulation Symposium , **ASS'2001**, Seattle, Washington (USA), April 22-26, 2001. Pages 185-192, ISBN 0-7695-1092-2.

Pere Vilà, Josep L. Marzo, "Scalability Study and Distributed Simulations of an ATM Network Management System based on Intelligent Agents", In proceedings of SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems, **SPECTS'2000**, pages 29-35, Vancouver (Canada), 16-20 July 2000. Edited by Mohammad S. Obaidat, Franco Davoli, Marco Ajmone Marsan, SCS ISBN 1-56555-206-7

Josep L. Marzo, **Pere Vilà**, Ramon Fabregat, "ATM network management based on a distributed artificial intelligence architecture", In proceedings of 4th International Conference on Autonomous Agents, **AGENTS'2000**, pages 171-172, Barcelona (Spain), 3-7 June 2000, Edited by Carles Sierra, Maria Gini, Jeffrey S. Rosenschein, ACM ISBN 1-58113-230-1

Pere Vilà, Josep L. Marzo, Ramon Fabregat, David Harle "A Multi-Agent Approach to Dynamic Virtual Path Management in ATM Networks", In proceedings of Communications Networks and the IMPACT of Software Agents, **IMPACT'99**, page 56 to 63, Seattle (USA) 2-3 December 1999. Edited by Alex L.G. Hayzelden ISBN – 0904188647.

Pere Vilà, Josep Lluís Marzo, Ramon Fabregat, "A Multi-Agent Approach to Dynamic Virtual Path Management in ATM Networks", In proceedings of Congrés Català d'Intel•ligència Artificial, **CCIA'99**, page 290 to 298, Butlletí de l'ACIA (ACIA Newsletter), no. 18-19, ISSN: 1577-1989, Girona (Spain) 25-27 October 1999.

Research Reports

Santiago Cots, **Pere Vilà**, Teodor Jové, "Management of Network Resources with Agents", **IIiA-UdG** Research Report Ref. 03-13-RR, Institut d'Informàtica i Aplicacions (Universitat de Girona), November 2003.

Pere Vilà, "Dynamic VP management in ATM (Asynchronous Transfer Mode) using intelligent software agents", **IIiA-UdG** Research Report Ref. 00-18-RR, Institut d'Informàtica i Aplicacions (Universitat de Girona), January 2000.

Other Publications

Publications not related with the work presented in this thesis.

Antonio Bueno, Ramon Fabregat, **Pere Vilà**, Jose Marzo, "On the Use of QoS Requirements as Base of Multicast Charging", In IEEE Global Communications Conference (**GLOBECOM 2003**), San Francisco (USA), December 1-5, 2003.

Jose Marzo, Eusebi Calle, Anna Urra, **Pere Vilà**, "Enhancing MPLS QoS routing algorithms by using the Network Protection Degree paradigm", In IEEE Global Communications Conference (**GLOBECOM 2003**), San Francisco (USA), December 1-5, 2003.

Liliana Carrillo, J. L. Marzo, **Pere Vilà**, "About the scalability and case study of AntNet Routing", In proceedings of Congrés Català d'Intel.ligència Artificial (**CCIA'2003**), Palma de Mallorca (Spain), October 22-24, 2003.

Liliana Carrillo, José L. Marzo, David Harle, **Pere Vilà**, "A Review of Scalability and its Application in the Evaluation of the Scalability Measure of AntNet Routing", In Proceedings of IASTED Communication Systems and Networks **CSN 2003**. Benalmádena, Spain. September 8-10 2003. Edited by C.E. Palau Salvador, ACTA Press, pp.317-323, ISBN 0-88986-388-1

Eusebi Calle, **Pere Vilà**, Jose L. Marzo, Santiago Cots, "Arquitectura del sistema de gestión de ancho de banda y protección para entornos de redes MPLS (SGBP)", In proceedings of Simposio de Informática y Telecomunicaciones, **SIT 2002**. September 25-27, 2002. Sevilla, Spain. Edited by R. Corchuelo, A. Ruiz-Cortés, and D. Sevilla. ISBN: 84-699-9417-4. Pages 143-154.

Antonio Bueno, Ramon Fabregat, **Pere Vilà**, "A Quality-Based Cost Distribution Charging Scheme for QoS Multicast Networks", In proceedings of 2nd European Conference on Universal Multiservice Networks (**ECUMN'02**), Colmar (France), April 8-10, 2002. ISBN 0-7803-7422-3.

Eusebi Calle, Teo Jové, **Pere Vilà**, Josep L Marzo, "A Dynamic Multilevel MPLS Protection Domain". In Proceedings of the 3rd International Workshop on Design of Reliable Communication Networks, **DRCN 2001**, Budapest (Hungary), 7-10 October, 2001. Edited by Tibor Cinkler.

José L. Marzo, Piergiulio Maryni, **Pere Vilà**, "Towards QoS in IP-based core networks. A survey on performance management, MPLS case", In Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems, **SPECTS'2001**, Orlando, Florida (USA), 15-19 July, 2001 Edited by Mohammad S. Obaidat, Franco Davoli, SCS ISBN 1-56555-240-7.

Teodor Jové, **Pere Vilà**, "A Teletraining Platform based on Web", In proceedings of Workshop on European Scientific and Industrial Collaboration **WESIC'98**, pp 253-258, Girona (Spain) 10-12 June 1998. Institut d'Informàtica i Aplicacions, ISBN 84-95138-08-5

José L. Marzo, Eusebi Calle, **Pere Vilà**, "QoS Protection: Formulation and experimental analysis of the MPLS case", **IIiA-UdG** Research Report Ref. 02-17-RR, Institut d'Informàtica i Aplicacions (Universitat de Girona), November 2002

Lluís Fàbrega, Teo Jové, José L. Marzo, **Pere Vilà**, "End-to-end admission control for a guaranteed minimum throughput service", **IIiA-UdG** Research Report Ref. 02-15-RR, Institut d'Informàtica i Aplicacions (Universitat de Girona), November 2002

Projects

January 2004 – December 2006
Participation in the project "Arquitectura multiservicio orientada a la fiabilidad y accesibilidad de redes troncales IP" Spanish Ministry of Science and Technology (MCYT) (ref. TIC2003-05567).

January 2003 – December 2004
Participation in the Thematic Network "Red temática de gestión de redes IP orientadas a circuitos vistuales (MPLS)". Spanish Ministry of Science and Technology (MCYT) (ref. TIC2002-10150-E).

January 2003 – December 2004
Participation in the Thematic Network "Xarxa temàtica de gestió de xarxes IP orientades a circuits virtuals (GMPLS/MPLS)". AGAUR – Catalan Government (ref. 2003/XT/00037).

September 2002 – March 2003
Participation in the project “Adaptive multiagent system for a web based tutoring environment”, in the EU funded network of excellence Agentcities (ref. Agentcities deployment grant cod 41. Member ship 14).

September 2000 – July 2002
Participation in project “Group for Advanced Learning Environments using Communication and Information Aids” (GALECIA). European Union SOCRATES-MINERVA project (ref 88089-CP-1-2000-1-PT).

January 2000 – December 2002
Participation in the project “Gestión Inteligente de redes orientadas a las nuevas aplicaciones telematicas con requerimientos de calidad de servicio” (Girona TRECS). Spanish Research Council (CICYT) (ref. TEL99-0976).

January 1999 – December 2000
Participation in the integrated action “Comparative study of dynamic routing in ATM networks”, Spanish Ministry of Education and Culture (ref. HI98-02). Joint project with the Department of Communications, Computer and Systems Science (DIST) - University of Genoa, Genoa (Italy)

September 1997 – September 1999
Participation in the project “Ensino a distância de informática” (EDIN). European Union MINERVA project (ref 39701-CP-2-98-1-PTA-ODL)