

Network Resource Management Investigation using a Distributed Simulator

Pere Vilà, Josep L. Marzo, Antonio Bueno, Daniel Massaguer
Institut d'Informàtica i Aplicacions (IiA) - Universitat de Girona
Lluís Santaló Av., 17071 Girona, SPAIN
phone: +34 972418475, fax: +34 972 418098
e-mail: { perev | marzo | bueno | dmassa }@eia.udg.es

Keywords: Distributed Simulation, Client/Server, Network Management and Control, MPLS, ATM.

Abstract

Network resource management deals with protocols and networks capable of performing a reservation of the available resources in order to guarantee a certain Quality of Service (QoS). Examples of these technologies are Asynchronous Transfer Mode (ATM) and Multi-Protocol Label Switching (MPLS), which are usually used in core networks. An important objective of network providers is to obtain the maximum profit from their resources, hence there is a need for an efficient resource management. There are different techniques and approaches in resource management: centralised, distributed, hybrid, based on artificial intelligence techniques, etc.

Investigation in this field is difficult, mainly because network research laboratories do not dispose of a large core network where they can investigate their approaches and algorithms. This paper presents a simple but flexible distributed simulator that supports a wide range of different experiments. It is based on an event-oriented simulation at a connection level (no packet or cell granularity). The distributed simulator is oriented to the simulation of large core networks and support different routing and admission control algorithms. It also presents a definition of the scenario where this simulator can be used, mainly in the context of Traffic Engineering, i.e. dynamic bandwidth management and fast restoration mechanisms. Some results in the dynamic bandwidth management area are presented.

INTRODUCTION

Although the technology of the present-day networks offers users increasing transmission capacity, the fact is that the increase in the amount of data to be transmitted is higher than the network capacity. This is due to the high increase in the number of new users and the appearance of new services (multimedia and interactive resource-consuming services). Therefore, it is clearly necessary to use network resources efficiently.

Large telecommunication companies have used powerful and centralised network management tools until now. These systems provide network statistics and powerful analysis tools to help human network managers in decision-making. These centralised tools suffer a scalability problem when excessive network monitoring traffic arises. Moreover, they are tools designed to perform manual network management or, at the most, to carry out some actions at pre-set hours of the day.

By using certain network management technologies the network adapts itself automatically under different levels of load in a dynamic way.

Generally network management encompasses a broad range of tasks that are summarised in the so-called FCAPS (Fault, Configuration, Accounting, Performance, and Security) management areas. The field of resource management deals mainly with Performance and Fault management, and more specifically with dynamic bandwidth management and fault protection mechanisms [1, 2].

This paper presents a distributed simulator to investigate network resource management. Simulation is a crucial tool in the investigation and evaluation of different mechanisms or techniques to perform this resource management. The investigation is focused on automatic tools, several of which are partially or totally distributed. The simulator itself is also distributed in order to facilitate the development and testing of these mechanisms.

The next section presents the background and the environment where these resource management techniques can be applied and details the main idea to perform this management, i.e. the logical network. It also details two network technologies that have mechanisms to establish and manage a logical network. Section 4 focuses on which are the specific actions that must be carried out by network management (i.e. fault protection and dynamic bandwidth management) and that the distributed simulator must support. Section 5 details the simulator design and its main characteristics. Section 6 presents a simulation scenario performed to evaluate a dynamic bandwidth management algorithm. Finally, section 7 presents the conclusions and future work.

BACKGROUND

There are several types of network technologies that dispose of dynamic resource management capabilities. These capabilities allow the design and implementation of automatic mechanisms to manage network resources. Usually the main resource that must be managed is the bandwidth; this requires the network technology to have some kind of hierarchical reservation capability, i.e. the ability to establish a logical network layer over the physical network. Then, the user connections are established through the paths of this logical network. The concept of a logical network or a logical topology is presented in figure 1.

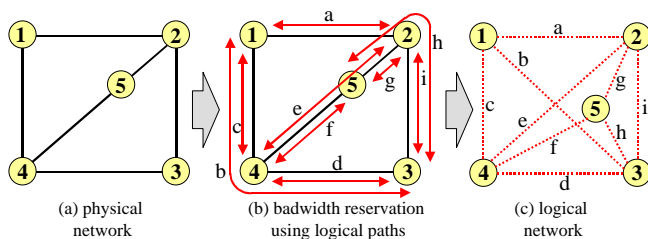


Figure 1. Example of a logical network established with the logical paths a, b, c, etc.

The logical paths can be seen as pre-reservations of bandwidth between different nodes in order to facilitate the establishment of user connections or flows.

The logical network can be dynamically changed and this implies that the network resources can be adapted to the traffic demand to obtain the maximum performance (and profit) from the available physical resources.

The logical network is usually also used to implement protection mechanisms, i.e. some of the logical paths can be established not to be used as working paths for the traffic demand but as a backup or protection paths in case of failure.

These mechanisms can be included in Traffic Engineering (TE) and they are detailed in section 4. The following points briefly describe two network technologies that make use of these hierarchical reservation mechanisms: MPLS and ATM. An example of an environment where to apply this mechanisms is presented in [3].

Multi-Protocol Label Switching (MPLS)

MPLS is a protocol that enables the mechanisms to manage the core network belonging to a network provider, usually in an Internet environment. MPLS groups user transmissions into flows and allows the allocation of a certain capacity to each flow [4, 5].

MPLS acts within a domain called MPLS domain. The 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, that packet is classified into a specific Forwarding Equivalent Class (FEC),

which groups the packets with certain common properties (protocol, size, origin, destination, etc.).

A label is assigned to every FEC and all the data packets belonging to the same FEC. While the packets are inside an MPLS domain, these packets go through pre-established paths called Label Switched Paths (LSP). The set of LSPs constitutes the logical network and it is established using a signalling protocol called Label Distribution Protocol (LDP) [6]. An example of an MPLS domain is depicted in figure 2.

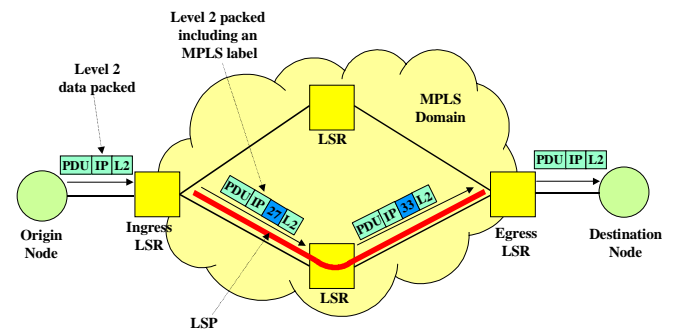


Figure 2. Example of an MPLS domain and its operation.

Asynchronous Transfer Mode (ATM)

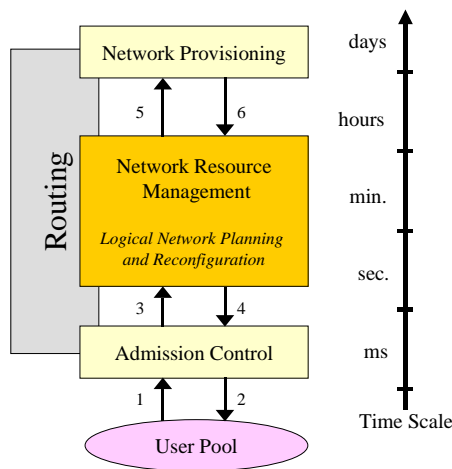
ATM networks are designed to support a wide range of services of diverse characteristics [7 - 9]. They have two layers of hierarchy: Virtual Path (VP) and Virtual Channel (VC).

Users can establish and release connections, i.e. VCs, through pre-established VPs. The Virtual Path layer is used to simplify the establishment of new connections and it also constitutes a logical network. This allows the network to carry out dynamic management of this logical topology and enables its adaptation to improve network resource utilisation [10, 11].

NETWORK RESOURCE MANAGEMENT EXPERIMENTATION

This section presents the main mechanisms that perform the network resource management. Usually these mechanisms act on a periodical basis, e.g. every hour. There are also restoration techniques that must re-route the links affected by a failure as fast as possible. Therefore, there are also temporal considerations since some mechanisms act in a short term, others in a mid-term and others in a long term. This time scale idea is depicted in figure 3.

The three main resource management functions supported by our distributed simulator are dynamic bandwidth management, fault protection and spare capacity planning. These functions are briefly described in sections 4.1 to 4.3.



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

Figure 3. Network resource management operations.

Bandwidth Management

Bandwidth management attempts to manage the capacities assigned to the different logical paths. Parts of the network can become under-utilised, and other parts congested. When this occurs, some connections are rejected which could be accepted if the traffic loads were better balanced.

One of the main objectives of bandwidth management is to minimise Call 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. Two actions are usually performed for the bandwidth management system: bandwidth re-allocation and logical path re-routing [10, 11].

If in the same link there are congested logical paths and under-used logical paths, the bandwidth assigned to each one can be re-allocated so that the worst call blocking probability in any given logical path is minimised. This method is called bandwidth re-allocation.

On the contrary, if all the logical paths in the link are congested or near congestion and there is not enough unutilised bandwidth for swapping between logical paths, then routes as well as capacities are altered to maximise the traffic carried on the network. In this case, a change in the logical network topology is required: if it is possible the logical paths can be redistributed in a better way to cope with the actual traffic demand.

Fault Protection

As networks should be fault-tolerant, rapid restoration after a failure is required. The ultimate goal is that customers do not notice failures. There are two main types of restoration schemes: dynamic and pre-planned. Pre-planned schemes are based on pre-assigned backup logical paths, whereas dynamic schemes are based on flooding algorithms and the search for restoration routes by messages broadcast after the failure is detected [12]. Pre-planned schemes restore effectively and rapidly, but require more spare resources [13].

A hybrid restoration is used when both schemes are simultaneously applied, i.e. some priority logical paths can be protected with pre-planned schemes and other lower-priority ones with dynamic schemes [14, 15].

Spare Capacity Planning

Network providers require high revenues: as bandwidth is an expensive resource, the objective is to minimise the bandwidth reserved for restoration procedures [16]. In other words, a good spare-capacity planning is crucial. A possible solution is the utilisation of hybrid restoration mechanisms, i.e. keeping sufficient spare capacity to protect all the VPs with backup schemes—an expensive solution. The spare capacity of the network should be carefully planned at the same time as the logical network and the protection backups. Therefore these actions modify the logical network and the spare capacity planning is also part of the resource management functions.

Interrelation of these techniques

Most of the above resource management techniques can be applied to different kinds of networks (ATM, MPLS, etc.) that have resource management mechanisms like the ones detailed in the previous section. These techniques are usually implemented using distributed and/or centralised algorithms, and sometimes a proposed system implements several of these techniques at the same time. All these techniques modify the logical path network, so they are very interrelated. For this reason there are several proposed methods that try to deal with all these techniques simultaneously, but this usually implies a high degree of complexity. To cope with this complexity there are many proposals for network resource management based on Distributed Artificial Intelligence (DAI) techniques, i.e. multi-agent systems.

The distributed simulator allows researchers to implement different resource management techniques regardless of whether they are centralised and/or distributed algorithms. Examples of distributed techniques are [17, 18], and examples of DAI based techniques are [19, 20]. The simulator even allows the development of non-automatic

resource management tools, i.e. manual utilities to manage the resources of the simulated network. An example of this kind of tools was developed using the previous simulator version [21] and it was able to manage the logical network through a powerful graphic user interface. In that example the user could establish, release and monitor every logical path of the entire network, and change its characteristics.

SIMULATOR DESIGN AND IMPLEMENTATION

The distributed simulator design is based on a client/server model [22]. The server process is called Node Emulator (NE) and its main function is the emulation of a working network node. This NE process offers a high functionality to its clients, which can be of two different types. The first type of client process is the Traffic Event Generator (TEG), that can be attached to an NE and acts as a pool of users asking connections or flows to a destination node. The TEG and NE sets of processes constitute the distributed simulator where resource management investigation is performed. The second types of client

processes are the network resource management algorithms. Both type of client processes, TEG and management, access to the server (NE) functionality through its Application Programming Interface (API). This API offers a set of functions to perform the node management, and allow the check and modification of the node internal status. The API also has several TEG-oriented functions, which are detailed later.

Performing a network simulation with a certain number of nodes implies the use of the same number of NE server processes, which can be in the same or in different computers. This is due to the use these processes make 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 [23], as well as in a PC under Linux OS [24].

Network resource management algorithms also have to be implemented as client processes. Thus it is possible to implement a centralised resource management algorithm or a distributed one, as it is shown in figure 4.

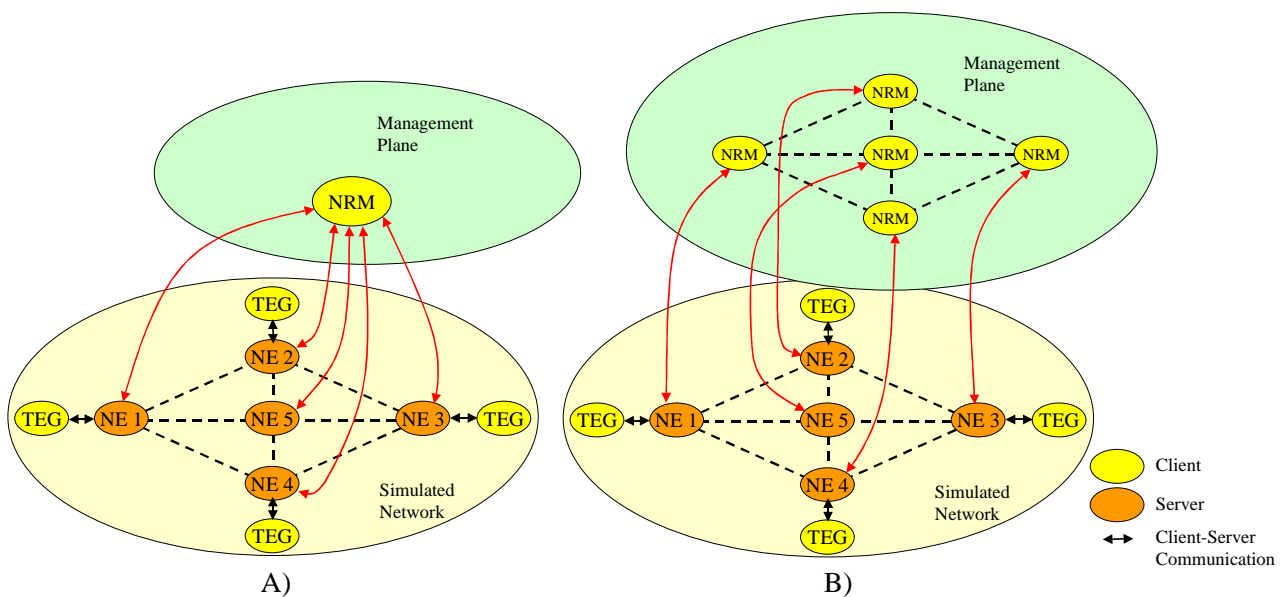


Figure 4. Distribution of client and server processes. A) Centralised management case. B) Distributed management case

Generally there is no communication among NE processes unless a specific routing protocol is implemented. The default routing protocol takes into account the direct path only. However, the default routing and admission control protocols can be changed through the modification of the NE source code. This is relatively easy to do due to its modular object oriented design, and its clear documentation.

The TEG processes are able to send events to the NE processes. The defined events include “connection-demand” and “connection-release” to establish new connections or flows and to release existing ones respectively. The generation of these events follows a parameterised distribution, e.g. a negative exponential distribution, thus the traffic load is independently configurable for every node.

Moreover it is possible to attach more than one TEG process to the same NE.

There are three additional types of events which can be generated by the TEG process. The first one is called “Instantaneous-BW” and for every established connection or flow, during its life span, the TEG process can generate this type of events to inform the NE of the real bandwidth used in that connection or flow. This type of events can also follow several different parameterised distributions. The second and third events are the “Link-failed-notice” and the “Link-restored-notice”, and they are used to inform an NE process to mark a physical link as failed or as a working link again. This is used to simulate link failures on the physical network. There is also a mechanism in the NE processes to send an alarm notification to the network resource management processes in case a link is marked as a failed link.

The different TEG processes are independent and all of them can be configured with different parameters. The TEG processes are not only used for the connection/flow distribution and connection/flow duration, but also to generate the different quality of service requested on each connection/flow and the selection of the destination node. These characteristics can also be set to configure, for example, a TEG to generate only one type of connection/flow with one specific destination node.

The simulated physical network is static and cannot change during the simulation. Moreover, all the NE processes read the physical network structure from the same configuration file. It is possible to set up an initial logical path network in the same common configuration file. If there is no kind of network resource management, the logical path network also remains static. The network resource management algorithms are not part of the simulation platform; they can be developed in any language on any platform and can be executed on the same or different computers from where the simulation processes are being executed. As it can be deduced, if the network resource management is implemented as a distributed algorithm, the communication among the management processes can use mechanisms or technologies different from the TCP/IP socket interface, e.g. CORBA [25] or Java RMI [26], allowing the implementation of many different types of management applications.

Every NE process generates a log file with all the functions and parameters that have been requested by the TEG(s) processes and/or the management process attached to it. From the analysis of this set of log files it is possible to extract 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 packets. Figure 5 shows an example of an NE log file. All the NE API functions have a numeric code and different number of parameters. In each line of the log file

there are the following fields: time in seconds from the start of the simulation, function code, the parameters values and the return value of that specific call. These values are separated by the character “*”. This file was created for the node 1 and, for instance, in this example the line “137.739894*11*2*2*****8” indicates that after 137.739894 seconds from the simulation starting, function 11 was called with two parameters both with value 2. The number 11 is the code for the “connection-demand” function and in this example a connection from node 1 (the node associated with this log file) to node 2 asking for a connection of type 2 (it is possible to define several different traffic classes for the connections). In this particular case the connection was accepted and the return value “8” indicates the connection identifier. The return value “0” means that the connection is rejected and the function 12 is the “connection-release” function.

```

TIME*FUNCTION*ARGUMENT[0]*ARGUMENT[1]*A
RGUMENT[2]*ARGUMENT[3]*ARGUMENT[4]*ARGU
MENT[5]*RESULT

47.137714*11*2*2*****1
57.337787*11*2*2*****2
57.937539*12*1*****
73.538251*11*2*2*****3
92.138668*11*2*2*****4
115.539242*11*2*2*****5
117.939269*11*2*2*****6
123.339338*11*2*2*****7
123.939622*11*2*2*****0
129.339765*11*2*2*****0
131.139421*12*3*****
137.739894*11*2*2*****8
143.140017*11*2*2*****0
144.940078*11*2*2*****0
146.140033*11*2*2*****0
153.939906*12*8*****
156.340015*12*7*****
.....etc.....

```

Figure 5. Fragment of a log file

From the above explanations it is possible to deduce the main characteristics of the distributed simulator. An important characteristic is its scalability in terms of the number of nodes that can be simulated. The simulation processes can be distributed in several computers and moreover they are executed in parallel. The main constrain is the client-server communication, but the use of an isolated network (e.g. 100Mbps Ethernet) allows the achievement of a relatively high performance.

EXAMPLE EXPERIMENT: DYNAMIC BANDWIDTH MANAGEMENT

This section describes a dynamic bandwidth management experiment in an MPLS domain. The scenario presents a simple four-node network, where several LSPs are established as is shown in figure 6. In this example, a

distributed algorithm based only on local information is tested. This distributed algorithm monitors the LSPs on every node. More specifically, there is a management process that monitors the LSPs that begin on a given node and calculate the Call Blocking Probability (CBP), i.e. the probability of a flow being rejected in the ingress LSR, for every LSP.

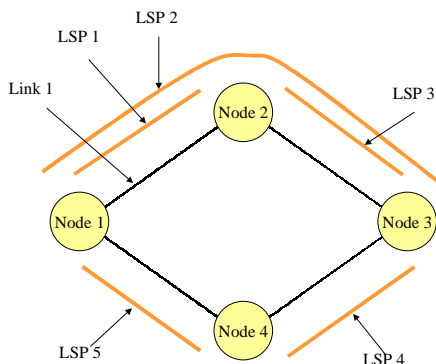


Figure 6. Four-node network with the established LSPs

If a certain CBP threshold is achieved, then the dynamic management process on that node asks the other involved nodes for bandwidth re-allocation, or if it is not possible for LSP rerouting. The necessary information is sent to other nodes by means of a developed application protocol. In this specific example the dynamic management and the monitoring processes were implemented using Java and interacted with the simulated nodes using their API (a Java package implementing an easy-to-use communication interface between an NE process and a Java application was also developed). The structure is presented in figure 7.

The distributed algorithm tested bases the decision of performing a logical network reconfiguration using only information that the node has locally. Moreover, no management process performs a reconfiguration of the whole logical network; each management process at each node only manages the resources on that node. Of course, once the decision is made, the management process on that node must co-ordinate the changes with the rest of the affected nodes.

Figure 8 shows the three different network situations in the course of the simulation. The main idea of this example experiment is that the flow demand from node 1 to node 2, which make use of the LSP1, constantly increases. This means that an LSP reconfiguration will be required to adapt the logical network topology to the new demand. First of all, the management processes try to expand the bandwidth assigned to an LSP without changing the logical network topology. This can be achieved using free bandwidth from the link and/or reducing the bandwidth of other LSPs that are

not full. In this particular experiment this action is performed for the first time when a CBP of 70% is reached on LSP1. After that, more flows can be accepted in LSP1, but as the user demand keeps increasing, once again a CBP of 70% for LSP1 is reached. In this case there is no more free bandwidth in link 1, and the management process decides to change the LSP2 and re-route it through node 4 instead of through node 2. Therefore, free bandwidth is released in link 1 and LSP1 capacity can be expanded to the maximum capacity of the link.

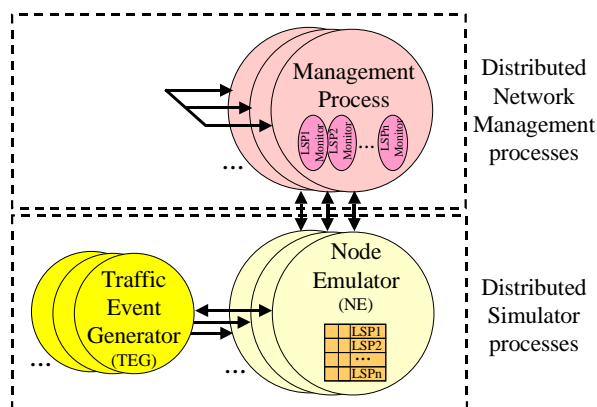


Figure 7. The different processes involved in the simulation example and the communication between them. On every Management process there are several threads each one monitoring an LSP.

This simple experiment shows how the management process at node 1 adapts the logical network in order to accept more flows from node 1 to node 2. Doing this dynamic resource management means that a better network performance is achieved, i.e. the same resources are better utilised. Without this resource management, in a static logical network scenario, more flows would have been rejected. Figure 9 also displays a graph including the LSP1 load, i.e. number of connections (left y-axis), and the LSP1 CBP (right y-axis) versus time.

Note that the demanded flows are homogeneous (2Mbps each) so before the first change, LSP1 can accept a maximum of 6 connections (LSP1=12Mbps). Between the first and second change LSP1 can accept 9 connections (LSP1=18Mbps) and after the second change it can accept 15 connections (LSP1=30Mbps). The LSP1 bandwidth changes are made when the CBP reaches 70% in this particular experiment.

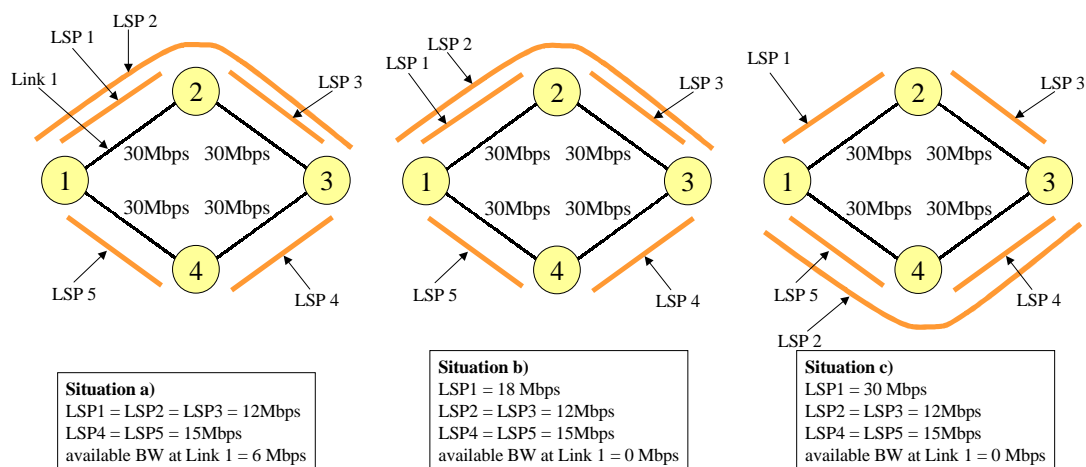


Figure 8. LSP network changes due to the bandwidth management mechanisms.

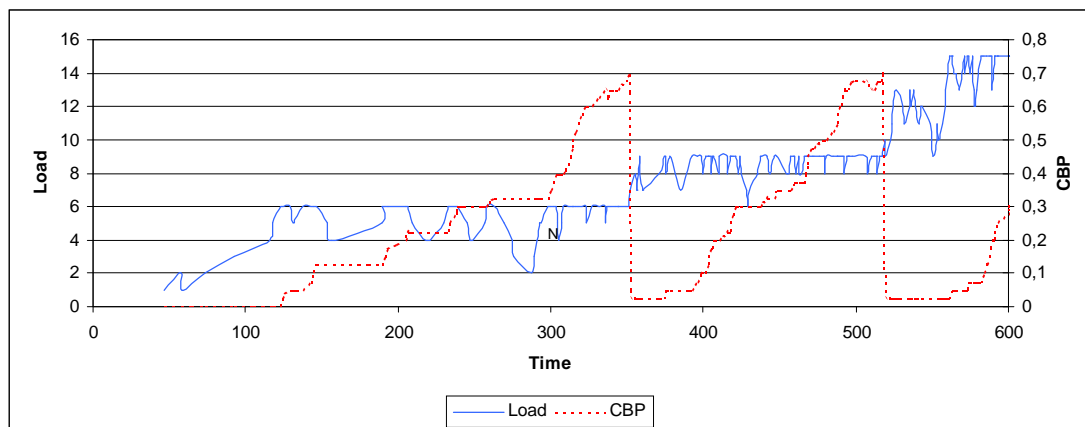


Figure 9. Load and CBP versus time for the LSP 1

CONCLUSIONS AND FUTURE WORK

This study has presented a distributed simulation platform able to perform a wide range of experiments related to network resource management. We believe there is a lack of network simulation platforms for network resource management where new algorithms or systems can be tested, because most general simulation platforms are control- and traffic-oriented, and usually centralised. The initial objective of designing a very flexible and configurable platform, while at the same time maintaining its modularity and simplicity, was also achieved by the use of client/server architecture.

At present, the simulation processes have been fully tested and are operative, and we plan to use it for a lot of experiments related to the research being carried out in our group, but we are now developing several network resource management mechanisms to be tested using the distributed simulator. These algorithms are still not fully implemented, so only simple examples and test simulations have been performed.

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 starting of a simulation. Secondly, we plan the development of an automatic tool to process the log files generated for every node. This tool could generate statistics and graphics which are always interesting and could also perform a merge of the several log files to extract more complex results. Finally these log files could also be used to perform a debug task of 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.

ACKNOWLEDGEMENTS

This work was partially supported by the Spanish Research Council (CICYT) under contract TEL-99-0976. The authors would like to thank the members of the BCDS group (University of Girona) for their useful comments.

REFERENCES

- [1] Sidor, D.J., 1998, "TMN Standards: Satisfying Today's Needs While Preparing for Tomorrow", IEEE Communications Magazine, (March).
- [2] Stallings, W., 1998, "SNMP and SNMPv2: The Infrastructure for Network Management", IEEE Communications Magazine, (March).
- [3] Marzo J.L., Maryni P., Vilà P., "Towards QoS in IP-based core networks. A survey on performance management, MPLS case", Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS'2001, Orlando, Florida (USA), 15-19 July, 2001.
- [4] Armitage G., "MPLS: The Magic Behind the Myths", IEEE Communications Magazine, January 2000.
- [5] Xiao X., Hannan A., Bailey B., Ni L.M., "Traffic Engineering with MPLS in the Internet", IEEE Network Magazine, March 2000.
- [6] Andersson L., Doolan P., Feldman N., Fredette A., Thomas B., "LDP Specification", RFC 3036, www.ietf.org
- [7] Kyas O., "ATM networks", International Thomson Computer Press, 1995, ISBN 1-85032-128-0.
- [8] Le Boudec J-Y., "The Asynchronous Transfer Mode: a tutorial", Computer Networks and ISDN Systems, vol 24, no 4, May 1992.
- [9] Sykas E.D., Vlamos K.M., Hillyard M.J., "Overview of ATM networks: functions and procedures", Computer Communications, vol 14, no 10, December 1991.
- [10] 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.
- [11] Sato K-I., Ohta S., Tokizawa I., "Broad-Band ATM Network Architecture Based on Virtual Paths", IEEE Transactions on Communications, vol 38 no 8, August 1990.
- [12] Song H-G., Lee H., Moon B., Chung S-J., "Dynamic Re-routing for ATM Virtual Path Restoration", IEEE GLOBECOM'97 Global Telecommunications Conference - Phoenix (USA), November 1997.
- [13] Yahia S.B., Robach C., "Self-Healing Mechanisms in ATM Networks: The Role of Virtual Path Management Functions", IEEE ICC'97 International Conference in Communications, June 1997.
- [14] Kawamura R., Ohta H., "Architectures for ATM Network Survivability and Their Field Deployment", IEEE Communications Magazine, August 1999.
- [15] Yahara T., Kawamura R., "Virtual Path self-healing scheme based on multi-reliability ATM network concept", IEEE GLOBECOM'97, November 1997.
- [16] 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.
- [17] Larsson S-O., Arvidsson A., "A Study of a Distributed Approach for VPC Network Management", IEEE GLOBECOM'97 Global Telecommunications Conference - Phoenix (USA), November 1997
- [18] Larsson S-O., Arvidsson A., "An Adaptive Local Method for VPC Capacity Management", ITC 16, P. Key and D.Smith (Eds.), Elsevier Science B.V., 1999
- [19] 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.
- [20] Marzo J.L., Vilà P., Fabregat R., "ATM network management based on a distributed artificial intelligence architecture", in 4th International Conference on Autonomous Agents, AGENTS'2000, Barcelona (Spain), June 2000.
- [21] Marzo J.L., Vilà P., Fàbrega L., Massaguer D., "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, IEEE Computer Society 2001.
- [22] Simulator URL: <http://brakali.udg.es/~simula>
- [23] Sun URL: <http://www.sun.com>
- [24] Linux URL: <http://www.linux.org>
- [25] CORBA URL: <http://www.corba.org>
- [26] Java URL: <http://java.sun.com>