An ATM Distributed Simulator for Network Management Research

Josep L. Marzo Pere Vilà Lluís Fàbrega Daniel Massaguer
Institut d'Informàtica i Aplicacions (IIiA)
Universitat de Girona, Girona, SPAIN
{marzo,perev,fabrega,dmassa}@eia.udg.es

Abstract

Due to the high cost of a large ATM network working up to full strength to apply our ideas about network management, i.e. dynamic virtual path (VP) management and fault restoration, we developed a distributed simulation platform for performing our experiments. This platform also had to be capable of other sorts of tests, such as Connection Admission Control (CAC) algorithms, routing algorithms, and accounting and charging methods.

The platform was posed as a very simple, event-oriented and scalable simulation. The main goal was the simulation of a working ATM backbone network with a potentially large number of nodes (hundreds). As research into control algorithms and low-level, or rather cell-level methods, was beyond the scope of this study, the simulation took place at a connection level, i.e. there was no real traffic of cells. The simulated network behaved like a real network accepting and rejecting calls and could be managed using standard tools, e.g. SNMP ones, or experimental tools using the API node.[1]

1. Introduction

There is a need for automated network management tools in large and complex networks, because human effort is insufficient. Network management is a huge field comprising: Fault, Accounting, Configuration, Performance and Security Management. Traditional network management systems are centralised approaches (Figure 1), with a scalability problem when the network grows. Typically, a network management system is a collection of tools for network monitoring and control that is integrated into a single operator interface with a powerful but user-friendly set of commands for performing most or all network management tasks.

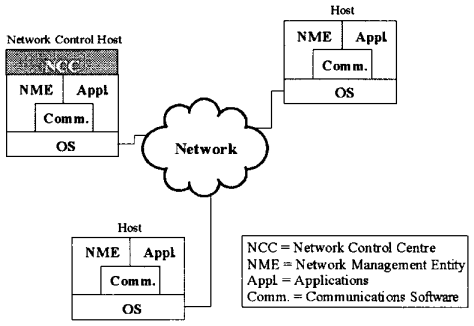

Figure 1: Centralised Network Management

The traditional architecture of a network management system is now briefly described. Each network node contains a collection of software devoted to the network management task, it isusually referred to as a Network Management Entity (NME) or Agent. Each NME performs the following tasks:

a) Collects statistics on communications and network-related activities

b) Stores statistics locally

c) Responds to commands from the network control centre, including commands to transmit collected statistics to the network control centre, change a parameter, provide status information and generate artificial traffic to perform a test.

At least one host in the network is designed as the network control centre (NCC) or Manager. Manager-Agent communication is carried out through an application-level network management protocol that uses the communications architecture in the same way as any other distributed application.

Well-known management standards include the Telecommunications Management Network (TMN) [21][15] and Simple Network Management Protocol (SNMP) [24][25]. More information on network management standards can be found in [3] and [19].

ATM networks are designed to support a wide range of services of diverse characteristics [12][13][27]. They have

[1] This study was partially supported by the CICYT (Spanish Education Ministry, under contracts TEL-98-0408-C02-01 and TEL-99-0976)

0-7695-1092-2/01 $10.00 © 2001 IEEE

two layers of hierarchy: Virtual Path (VP) and Virtual Channel (VC) levels (see Figure 2).

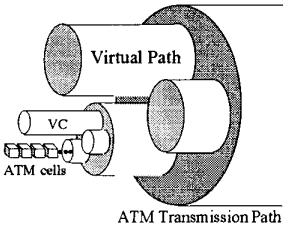

Figure 2: ATM hierarchy

Users can establish and release connections, i.e. Virtual Channels, through pre-established VPs. The Virtual Path layer is used to simplify the establishment of new connections and also constitutes a virtual topology over the physical network (see Figure 3). This allows dynamic management of this virtual topology and its adaptation to improve network resource use [5][20].

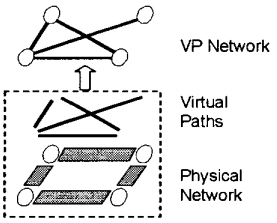

Figure 3: Virtual Path Network

Since one of our research topics is the dynamic management of the VP network, we initially planned simulation to perform this concrete task. But we found that a more general tool could be designed to perform other kind of experiments we were interested in. The following is an example of the kind of experiment we want to perform on our simulation platform. The three main VP management functions, bandwidth management, fault restoration and spare capacity planning, are described as follows.

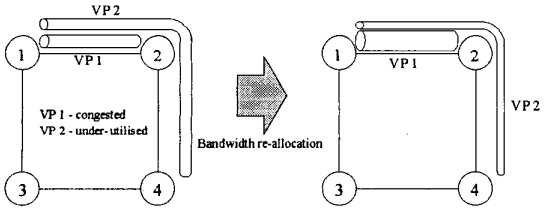

Figure 4: Bandwidth re-allocation or tuning

Bandwidth management attempts to manage the capacities assigned to the different VPs. 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 load were better balanced. A connection will be rejected by the network when the capacity reserved for the VP it is to traverse has been used by existing connections so much that there is not enough capacity left for the new connection.

The main objective is to minimise Call Blocking Probability (CBP), i.e. the probability that a call offered has of being rejected due to not enough capacity being available for allocation of the new call. Two actions are usually taken for the bandwidth management system:

a) If in the same link there are congested VPs and under-used VPs, the bandwidth assigned to each VP can be reconfigured so that the worst call blocking probability in any given VP is minimised. This method is called bandwidth re-allocation (Figure 4).

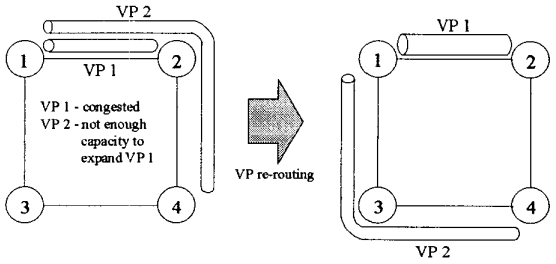

Figure 5: VP re-routing or topology tuning

b) If all the VPs in the link are congested or near congestion and there is not enough unutilised bandwidth capacity for swapping between VPs, then routes as well as capacities are altered to maximise the traffic carried on the network. In this case, a change in VP network topology is required: the VPs can be redistributed in the best possible way to cope with the actual traffic demand. In other cases, the priority is to minimise the number of re-routed VPs so that the minimum number of connections are affected, in which case the problem becomes a routing one (Figure 5).

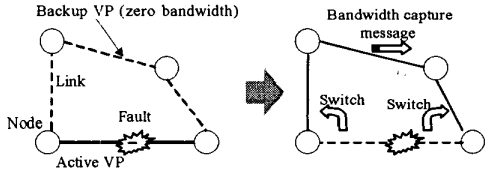

Figure 6: Pre-planned restoration

As networks have to be fault-tolerant, rapid restoration after a failure is required. The ultimate goal is that customers do not perceive failures. There are two main types of restoration schemes: dynamic and pre-planned. The latter restores effectively and rapidly, but requires more spare resources [28][34]. Pre-planned schemes (Figure 6) are based on pre-assigned backup VP, whereas dynamic schemes are based on flooding algorithms and

the search for restoration routes by messages broadcast after the failure is detected.

In hybrid restoration both schemes are applied, i.e. some priority VPs can be protected with pre-planned schemes and other lower-priority ones with dynamic schemes [33][11].

In this framework, a third problem arises, that of spare capacity planning. The introduction of hybrid restoration mechanisms is basically to save on spare capacity, i.e. the need for spare capacity to protect all the VPs with backup schemes is a very expensive solution. Network providers want economic benefits: as bandwidth is an expensive resource, the objective is to minimise the bandwidth reserved for restoration procedures [32]. In other words, good spare-capacity planning is required.

Recent studies propose different solutions to these network management problems. Most of these studies agree that some management tasks cannot be centralised after the network reaches a certain size, and that these management tasks should be automated and distributed round the network. One of our research lines focused on this task by using distributed artificial intelligence, i.e. intelligent agents [16][29][30].

For daily network management, use of intelligent software agents is proposed. These are defined as software entities with special properties (autonomy, social ability, reactivity and pro-activeness) [18][31]. The main objective is to automate these management functions and so improve network performance.

As network management is a good field for applying these techniques, several Multi-Agent approaches to these various network management problems have been developed [1][2][4][7][8][10][14][22][23]. Concretely, the work presented in [9] gives an overview of intelligent agents in telecommunications.

The following section describes simulation platform architecture and its main objectives and characteristics. Section 3 briefly presents the details of simulator platform development and Section 4 presents the initial experiences using our simulation platform. Finally, conclusions and future work are described.

2. Simulator Design

As simulator design is based on a distributed system, where multiple processes act as nodes and clients, the general architecture is described first and then each process is detailed.

Each ATM node is emulated by a process called ATM Switch Emulator (ASE) and so, to simulate a network with e.g. 20 nodes, the same number of ASE processes is needed. Nodes communicate at the TCP/IP level using the socket interface, whereby these processes can be executed on the same or different computers. In fact, the idea is to use our research network based on several PCs, using Linux OS, and interconnected by a local area network.

The physical network and its characteristics are defined by use of a common configuration file for all ASEs. Each ASE contacts with another when a physical link is established between them. Therefore, communication is not performed among all the nodes, but TCP/IP sockets only communicate each ASE with its neighbours, achieving the system scalability we sought.

Another basic process is the Traffic Event Generator (TEG). This process is completely configurable and capable of sending events to the different ASE nodes asking for new connections with certain characteristics, releasing connections, etc. It is possible to execute a different TEG process for each ASE in the simulation but not necessary, i.e. a network node can be configured as edge node or transit node of the core network. ASE-TEG communication also uses TCP/IP sockets.

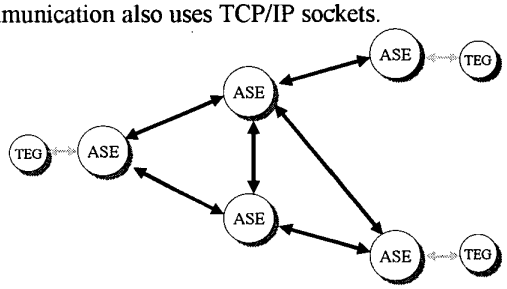

Figure 7: Example network with 5 nodes

Figure 7 shows a simulation example with 5 nodes. Three nodes represent edge nodes of a core network and two nodes represent transit nodes. Therefore, 5 ASE processes and 3 TEG processes are needed. The lines between processes represent the communication channels between them, but it is possible to understand the lines between ASE processes as a representation of the physical network defined in this example. Note that the processes can be executed on one or on several computers.

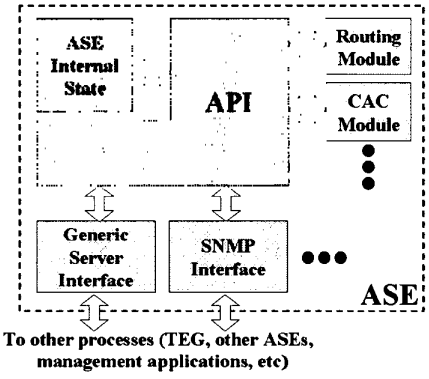

Figure 8: ASE internal structure

The ASE internal design is based on independent modules (Figure 8). The ASE core modules are the internal state module, which is in charge of storing all the internal data (variables and tables) of the system, and the API module, which is the only way to access to the internal state module. One of the main characteristics of the API module is that it has a complete set of procedure calls to access and manipulate all the ASE functions (Table 1).

All these basic functions can easily be extended by adding new modules. For example, modules with high interaction (through the API) with the internal state, such as CAC and routing modules, can be added by replacement of the default modules. The default CAC module uses the simple technique of the equivalent bandwidth algorithm, and the default routing module uses a simple static table and only takes into account the direct path. These modules are used by the ASE functions such as "VCsetup" to establish new virtual channels. Modules can be replaced by more powerful user-defined modules, just by following the given architecture, function names and parameters.

It is also possible to add another type of module, less interrelated with ASE internal functions and more interrelated with network management. We defined a general server interface to interconnect and communicate all ASE and TEG processes. Access to all API functions is possible through this interface, which uses TCP/IP sockets. In consequence, an external application can manage and monitor the simulated ATM network.

Table 1: ASE functionality

TEG Events	ConnectionDemand	Event asking for a new virtual channel with certain characteristics
	ConnectionRelease	Event informing about a virtual channel release
	InstantaneousBW	Event informing about the actual use of a concrete virtual channel
VP Management	Vpdemand	Function to ask if there is enough room for a new VP
	Vpsetup	Function to establish a new virtual path
	Vprelease	Function to release an existing virtual path
	VpavailableBW	Function to find the available bandwidth of a given virtual path
	CurrentVPLoad	Function to find the real bandwidth used in a virtual path
	VPReset	Function to release all the virtual channels of a given virtual path
	ChangeBW	Function to change the bandwidth assigned to a given virtual path
VC Management	Vcdemand	Function to ask if there is enough room for a new VC on a given VP
	Vcsetup	Function to establish a new virtual channel
	Vcrelease	Function to release an existing virtual channel
Monitoring and log	GetTables	To get all the relevant data from the ASE internal state
	NewLogFile	To close the actual log file and generate a new log file
Debug	ShowTables	To obtain all the data and variables of the ASE internal state
Fault simulation	LinkFault	Function to mark a physical link with a failure
	LinkRepaired	Function to restore a failed physical link
	Vpfault	Function to mark a virtual path with a failure
	Vprepaired	Function to restore a failed virtual path
	NodeFault	Function to mark a node with a failure
	NodeRepaired	Function to restore a failed node

Table 2: TEG events

ConnectionDemand	The TEG process asks the ASE for a new virtual channel from the node where the TEG is connected to another network node, with certain characteristics of traffic type and quality of service. After this the TEG awaits the ASE response. If the connection is accepted the TEG writes down the connection identification and the time assigned so as to know when to release it. If the connection is rejected, no other action for this connection is performed.
ConnectionRelease	When it is time to release a previously opened connection, the TEG sends this event to the ASE node which it is connected to, with its connection identification. TEG waits for the ASE confirmation that the virtual channel is released.
InstantaneousBW	While a connection lasts, this event can be used to inform about the bandwidth that is really used. When a new connection is requested, a new VC with a certain capacity is established, which makes it possible to report continuously what concrete percentage of the connection's assigned capacity (e.g. 90%) is being used at any given moment.

For instance, a new communication interface module able to accept requests by using SNMP/UDP can also be defined. Then this module can be seen as an SNMP agent with its Management Information Base (MIB), which enables our simulated network to be queried with standard network management tools.

TEG design has taken into account the goal of building up a powerful and completely configurable event generator. TEG is able to generate three different types of events (Table 2). These events are sent to the ASE to which the TEG is connected (communication is through the ASE Generic Server Interface using TCP/IP sockets).

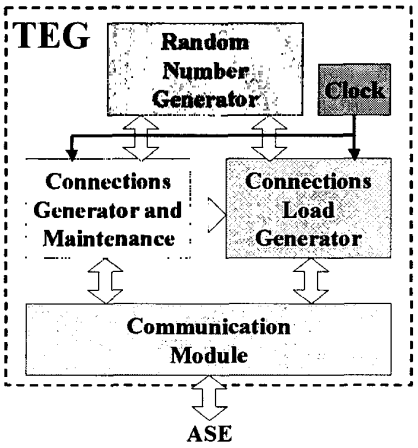

Figure 9: TEG internal structure

The TEG design is also modular. One module is a powerful random generator capable of generating random numbers following a certain distribution. There are two more modules, a module following the connections (i.e. the requests and the release of connections) and another following each established connection and generating the InstantaneousBW events. Both are based on the maintenance of dynamic lists with timestamps of when the different events have to be sent. The internal structure of the TEG process is shown in Figure 9.

The different TEGs are not synchronised and all of them can be configured with different parameters. Random numbers are used not only for the time distribution of the three events shown above, but also to generate the different qualities of service requested, and select the destination node on the network. These characteristics can also be set to configure, for example, a TEG to generate only one type of connection with one specific destination node.

3. Simulator Development

The simulation platform (basically the ASE and TEG processes) was developed in C++ under Linux. The main goal of this module is the modularity of the system, the ease of re-arrangement of new software pieces (e.g. new versions of CAC or routing algorithms), and its adaptability to different experiments. These goals are achieved through the following techniques:

a) The use of configuration files: users can produce a set of configuration files, each defining a particular experiment. To perform a new experiment, users need only change the configuration file, where the system reads the corresponding parameters, and start the simulation.

b) The API interface guarantees total access to the ASE's functions. It is easy to add or change modules and re-compile the application.

c) The ASE server structure ensures communication with any type of client defined according to our communication protocol, or any other protocol using the appropriate interface (e.g. we planned to build a standard SNMP interface).

d) The log-book file for writing results. Each ASE maintains a log file where every internal modification of tables, system variables, and events are recorded. This ensures complete control of the experiment's results without the huge amount of data that would be needed to save the system's data at regular time intervals. It is also possible to program a client procedure, which queries all the ASE processes at regular intervals to obtain the desired data.

We have not yet evaluated system performance. This is not a major goal right now: we are pleased enough with the performance achieved so far. We have only tested our system on an isolated (i.e. without any other type of network traffic) local area network comprising four full-time personal computers running the operating system and the simulation platform. Furthermore, we expect an easy adaptation of the platform to different parallel execution environments such as parallel virtual machine (PVM) environments [6].

4. Test and Experiments: The AMA Application

To test the platform, a network management application was developed and connected to the simulation platform. This application was designed as a centralised network management application with a powerful graphic-interface that allows a network manager to monitor, obtain statistics and manage the simulated ATM network. This is called the ATM Management Application (AMA).

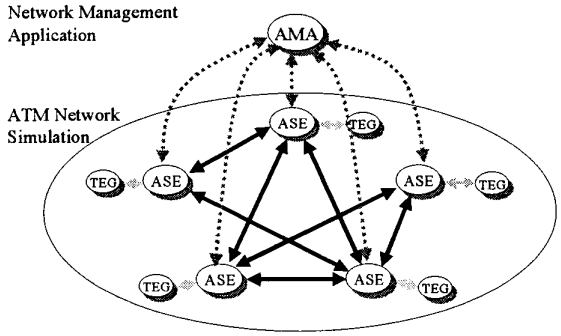

Figure 10: ATM Management Application (AMA) framework

The AMA is connected to each ASE on the platform, collecting data from the ASE processes. From the distributed view of each ASE, it gets its general and centralised view of the simulated network and so gives an intuitive view to the network manager (Figure 10).

AMA knows nothing about the network itself except the information obtained from each connected ASE. AMA only uses one configuration file to know the node addresses of the various ASEs.

AMA is divided into three main modules: management and configuration, monitoring, and generation of statistics, which are explained below.

As the simulation is performed at the connection level, most AMA management and configuration functions are oriented to Virtual Path management. These functions allow the user to:

The AMA is connected to each ASE on the platform, collecting data from the ASE processes. From the distributed view of each ASE, it gets its general and centralised view of the simulated network and so gives an intuitive view to the network manager (Figure 10).

AMA knows nothing about the network itself except the information obtained from each connected ASE. AMA only uses one configuration file to know the node addresses of the various ASEs.

AMA is divided into three main modules: management and configuration, monitoring, and generation of statistics, which are explained below.

As the simulation is performed at the connection level, most AMA management and configuration functions are oriented to Virtual Path management. These functions allow the user to:

a) Add and delete VPs dynamically by changing the logical network configuration on line and adapting it to the traffic demand.

b) Reset VPs. The network manager can delete all the Virtual Channels that go through a Virtual Path. This helps achieve a stable state once an experiment is finished, so that the platform is ready to start another experiment (with the given new physical and logical topology).

c) Simulate faults on the physical links and on the Virtual Paths. This function allows the network management to mark physical links or Virtual Paths as if they had failed. Therefore, the network manager can evaluate how the restoration mechanisms act.

d) Monitor the network status. AMA allows users to monitor the bandwidth assigned and used by a physical link as well as a VP, and also the bandwidth used by each VC. It is also possible to monitor the number of connections requested and accepted in the whole network, at each node and at each active VP.

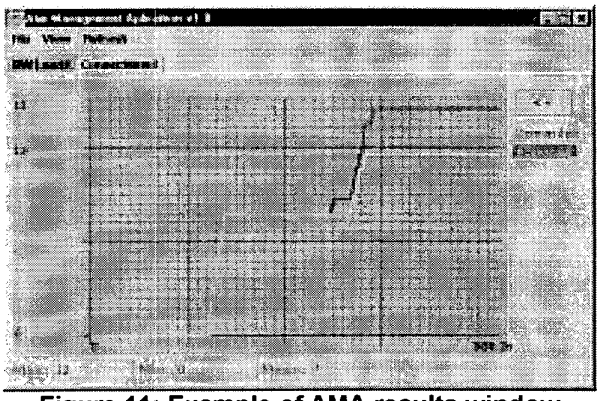

Figure 11: Example of AMA results window

e) Present simplified statistics. AMA requests and stores the network nodes' data (the ASE data) at configurable regular time intervals. This enables the load (i.e. capacity assigned and used) of the simulated network versus time (per VP, VC, physical link, or the whole network) to be calculated. It also enables the requested and accepted connections over time (per VP, node or whole network) to be shown. These statistical results allow the evaluation of the blocking probability, defined as the ratio between the connections not accepted and all the connections demanded. Figure 11 shows the requested and accepted connections versus time statistic.

Figure 12 shows two windows captured by the AMA application. Figure 12a shows the physical network topology while a physical link is monitored in the smaller window. Figure 12b shows the logical network (the established VP) while the requested and accepted connections are being monitored in the smaller window. Note that lines joining two nodes in the logical network representation may represent one or more VPs between these two nodes.

AMA has been developed as a Java 2 application, and so can be executed on any computer able to run the Java Virtual Machine v.1.2 [17][26]. Therefore, any computer

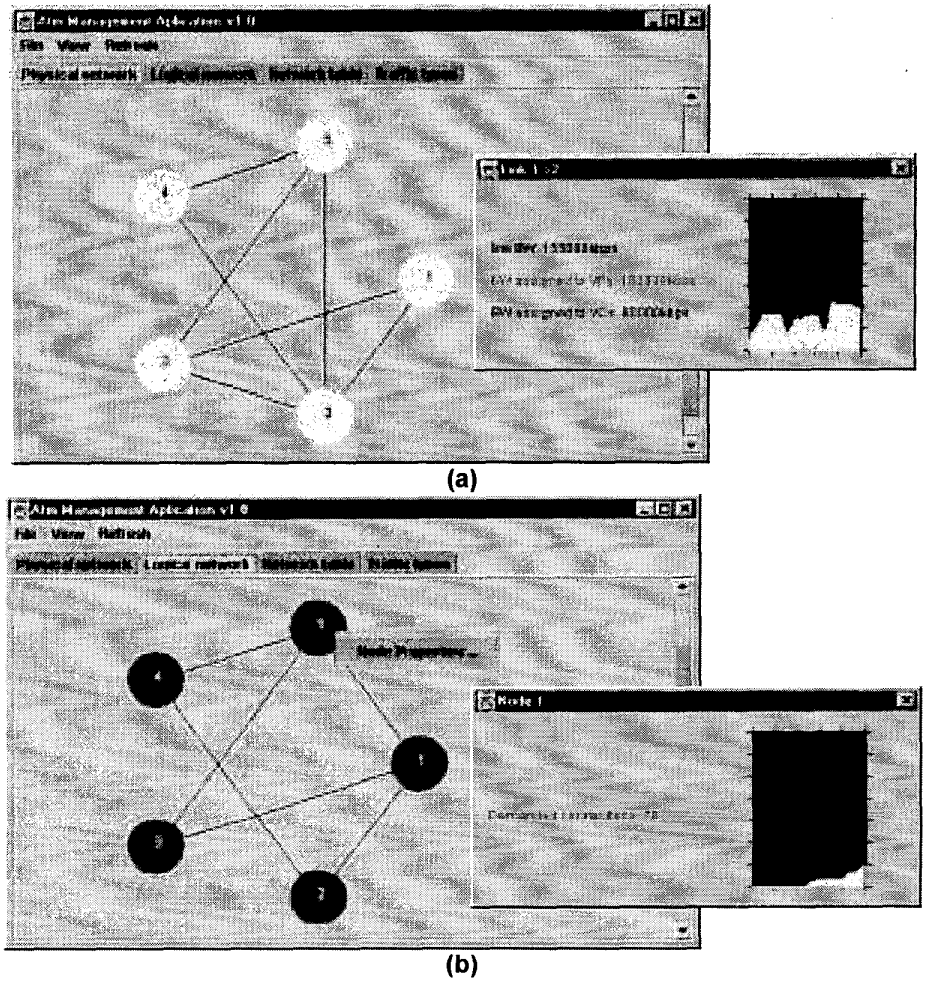

Figure 12: AMA window captures

connected to Internet can manage the simulation. Only one computer at a time can control the simulation: a graphic and an intuitive interface are then provided. AMA communicates with each ASE through TCP/IP sockets and uses ASE Generic Server Interface to obtain information from each ASE and to manage it.

5. Conclusions and Future Work

This study has presented a distributed simulation platform able to perform a wide range of experiments related to network management. We believe there is a lack of network simulation platforms for network management where new tools can be tested, because most general simulation platforms are control- and traffic-oriented, and usually centralised. Our initial objective of designing a very flexible and configurable platform, but maintaining its modularity and simplicity, was also achieved. It is also possible to use standard network management tools and protocols without too much modification or adaptation to our platform, which permits comparison of different management techniques using the same simulated network in a simple way.

At present, we have just finished the implementation of the basic modules of our simulation platform. First, the platform needs to be exhaustively tested while at the same time more modules can be developed. When the platform has been fully tested and is operative, we plan to use it for a lot of experiments related to the research being done in our group. Finally, as we have developed our ATM-oriented platform, future research could study the extension of our platform to simulate other kinds of networks, such as backbone IP networks. This will allow us to perform experiments on new IP techniques and protocols we are also interested in (e.g. IP version 6 or

Multi-Protocol Label Switching –MPLS) and to research the management of this type of network.

6. References

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

[2] Bigham, J., Cuthbert, L.G., Hayzelden, A.L.G., Luo, Z., 1999b, "Flexible Decentralised Control of Connection Admission", Proceedings of IMPACT'99, Seattle, USA (December).

[3] Black, U., 1994, "Network Management Standards", 2nd ed., McGraw Hill.

[4] Davison, R.G., Hardwicke, R.G., Cox, M.D.J., 1998, "Applying the agent paradigm to network management", BT Technology Journal, (July).

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

[6] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Mancheck, R., Sundram, V., 1994, "PVM 3 User Guide and Reference Manual", US Department of Energy, Oak Ridge Tenesee, e-mail: pvm@msr.epm.ornl.gov

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

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

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

[10] Hayzelden, A.L.G., Bigham, J., Luo, Z., 1999, "A Multi-Agent Approach for Distributed Broadband Network Management", 4th International Conference and Exhibition on Practical Application of Intelligent Agents and Multi-Agents, London, UK, (April).

[11] Kawamura, R., Ohta, H., 1999, "Architectures for ATM Network Survivability and Their Field Deployment", IEEE Communications Magazine, (August).

[12] Kyas, O., 1995, "ATM networks", International Thomson Computer Press.

[13] Le Boudec, J.Y., 1992, "The Asynchronous Transfer Mode: a tutorial", Computer Networks and ISDN Systems,vol 24, no 4, (May).

[14] Lucent Technologies White Paper, 1997, "Proactive Problem Avoidance and QoS Guarantees for Large Heterogeneous Networks", Lucent Technologies, Rensselaer Polytechnic Institute, and Pennsylvania State University. URL:http://www.cs.rpi.edu/~kaploww/finalproposal.html

[15] Manley, A., Thomas, C., 1997, "Evolution of TMN Network Object Models for Broadband Management", IEEE Communications Magazine, (October).

[16] Marzo, J.L., Vilà, P., Fabregat, R., 2000, "ATM network management based on a distributed artificial intelligence architecture", Accepted in 4th International Conference on Autonomous Agents, AGENTS'2000, Barcelona, Spain (June).

[17] Morgan,M., 1998, "Using Java 1.2", Que Corporation ISBN 0-7897-1627-5

[18] Nwana, H.S., 1996, "Software agents: an overview", The Knowledge Engineering Review Vol.11 No.3.

[19] Raman, L., 1998, "OSI Systems and Network Management", IEEE Communications Magazine, (March).

[20] Sato, K.I., Ohta, S., Tokizawa, I., 1990, "Broad-Band ATM Network Architecture Based on Virtual Paths", IEEE Transactions on Communications, vol 38 no 8, (August).

[21] Sidor, D.J., 1998, "TMN Standards: Satisfying Today's Needs While Preparing for Tomorrow", IEEE Communications Magazine, (March).

[22] Somers, F., 1996, "HYBRID: Unifying Centralised and Distributed Network Management using Intelligent Agents", IEEE/IFIP NOMS'96, Network Operations and Management Symposium, (April).

[23] Somers, F., Evans, R., Kerr, D., O'Sullivan, D., 1997, "Scalable low-latency network management using intelligent agents", ISS'97, World Telecom Congress, (September).

[24] Stallings, W., 1996, "SNMP, SNMPv2 and RMON: Practical Network Management", Addison-Wesley, ISBN 0-201-63479-1

[25] Stallings, W., 1998, "SNMP and SNMPv2: The Infrastructure for Network Management", IEEE Communications Magazine, (March).

[26] Sun Microsystems 1999, "Java 2 SDK – Standard Edition Documentation, version 1.2.2-00.1", http://java.sun.com/products/jdk/1.2/docs/index.html

[27] Sykas, E.D., Vlakos, K.M., Hillyard, M.J., 1991, "Overview of ATM networks: functions and procedures", Computer Communications, vol 14, no 10, (December).

[28] Veitch, P.A., 1996, PhD thesis "Virtual Path Restoration Techniques for Asynchronous Transfer Mode Networks", University of Strathclyde.

[29] Vilà, P., Marzo, J.L., Fabregat, R., Harle, D., 1999, "A Multi-Agent Approach to Dynamic Virtual Path Management in ATM Networks", IMPACT'99, Seattle, USA (December).

[30] Vilà, P., Marzo, J.L., 2000, "Scalability Study and Distributed Simulations of an ATM Network Management System based on Intelligent Agents", SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS'2000, Vancouver, Canada (July).

[31] Wooldridge, M., Jennings, N.R., 1995, "Intelligent Agents: Theory and Practice", Knowledge Engineering Review, (January).

[32] Xiong, Y., Mason, L.G., 1999, "Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks", IEEE/ACM Transactions on networking, (February).

[33] Yahara, T., Kawamura, R., 1997, "Virtual Path self-healing scheme based on multi-reliability ATM network concept", IEEE GLOBECOM'97, (November).

[34] Yahia, S.B., Robach, C., 1997, "Self-Healing Mechanisms in ATM Networks: The Role of Virtual Path Management Functions", ICC'97 International Conference in Communications, (June).