

# Taming Xunet III

***Nikos G. Aneroussis, Aurel A. Lazar and Dimitrios E. Pendarakis***

Department of Electrical Engineering  
and  
Center for Telecommunications Research  
Columbia University, New York, NY 10027-6699  
e-mail: {nikos,aurel,dimitri}@ctr.columbia.edu

## Abstract

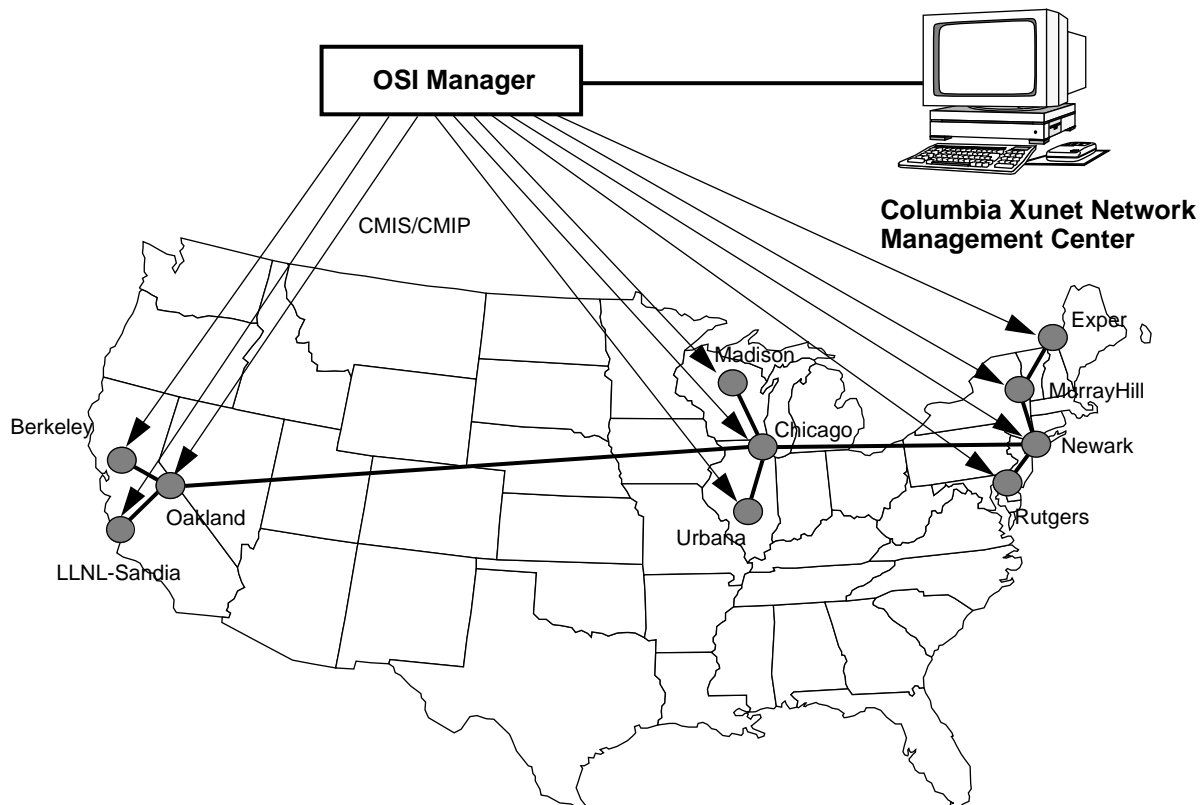
An architecture for network management and control for emerging wide-area ATM networks is presented. The architecture was implemented on XUNET III, a nationwide ATM network deployed by AT&T. The Xunet network management system is based on the OSI standards and includes configuration, fault and performance management. An OSI agent resides at every switching node. Its capabilities include monitoring of cell level quality of service in real time and estimation of the schedulable region. The complexity and accuracy of real-time monitoring functionalities is investigated. To provide realistic traffic loads, distributed traffic generation systems both at the cell and call level have been implemented. In order to study the trade-off between the network transport and signalling system we have implemented a virtual path signalling capability. Our experiments show that the ability of a network to admit calls is limited by two distinct factors: the capacity of the network and the processing power of the signalling system. Depending on the bandwidth requirement of calls, the limit of one or the other will be first reached. This is a key observation, unique to broadband networks.

## 1. Introduction

---

Xunet is one of the five Gigabit testbeds (also known as the BLANCA testbed) sponsored by the Corporation for National Research Initiatives. It has been deployed by AT&T in collaboration with several universities and research laboratories in the United States [FRA92]. Figure 1 shows the topology of the backbone network. Xunet has two principal hardware components: routers and switches. Transmission links between the switches are currently rated at 45 Mbps and are gradually substituted with 622 Mbps optically amplified lines. Each router attaches a local area network to the backbone network. The router, based on a Silicon Graphics workstation, has two interfaces: one to a local FDDI ring; the other through a 200Mbps ATM trunk to the switch (see also Figure 4). A variety of standard network interfaces (TAXI, HiPPI) will be available in the near future. A Silicon Graphics workstation serves as the switch Control Computer at each network node. The Control Computer runs the switch control software that performs signalling, control and fault detection functions. It is connected to an Xunet switch through an Ethernet interface.

Xunet is an ideal broadband networking platform for experimentation, not only for high-bandwidth networked applications, but also for the evaluation of software experiments that will enable the deployment of the emerging ATM networks. For example, Xunet can be used to evaluate algorithms for connection management as well as admission control, scheduling, flow control and buffer management.



**Figure 1:** The Xunet Topology and Management System

In this spirit, we have started two projects that we believe are central to taming the complexity and ensuring the smooth and efficient operation of ATM networks. The first project has the objective of defining a network management architecture for ATM networks and will be described in Section 2. Section 3 presents the second project that aims at developing and evaluating algorithms and software for the efficient utilization of network resources. In Section 4, results of network experiments on the cell and call performance are presented.

The combined work in the two projects allows us to load, monitor, control and evaluate a broadband network that spans the continental United States in a very simple and flexible manner from a single workstation.

## 2. The Xunet OSI Management System

The summer of 1992 we started, using Xunet as a testbed, the development of a network management system based on the then evolving OSI standards for network management [ANE93]. The challenges were many: Firstly, since Xunet is an ATM network, the design of the management system is fundamentally different from current SNMP-based management systems for IP-based networks. Secondly, little was known at that time on the implementability of the OSI management standards on any given platform, and, in addition, the number of available OSI managed systems was extremely limited.

## 2.1 The OSI Management Model

The OSI management model is a client/server or manager/agent interaction model [ISO88]. According to this model, network entities (either physical, like hardware modules, or logical, like virtual circuits) are mapped into managed objects for monitoring and control purposes. The managed objects are also referred to as *logical objects* and the network entities that they represent as *real objects*. A Management Agent contains the information about the managed objects in the Management Information Base (MIB). The MIB is an object-oriented database. Managed objects are characterized by a set of attributes that reflect the state of the corresponding real object and behavioral information, that defines the result of management operations on the managed object. A proprietary protocol can be used for *linking* the state of every real object to its logical counterpart in the MIB. The Manager connects to the agent(s) and performs operations on the objects in the MIB using CMIP (the Common Management Information Protocol). These operations are of synchronous nature, i.e., they are initiated by the manager who, then, waits for a reply from the agent(s). Events of asynchronous nature (notifications) such as hardware faults can be emitted from the agent(s) to the manager using the event reporting primitive of CMIP.

## 2.2 Goals of the Xunet Management System

We have set the following goals for the management project on Xunet:

1. Identify the structure of the required management system for an ATM network. At the agent side this involves the definition of the structure and behavior of Managed Objects, the Inheritance and Containment Hierarchy, etc. At the manager side this involves the definition of the scope and functionality of management applications, the appropriate user interfaces, etc.
2. Investigate the implementation issues (expandability, fault tolerance, state consistency between the logical and the real objects, methods of communication between them, etc.)
3. Based on the experience with the prototype management system, propose integrated management solutions for emerging ATM networks. At the same time propose a standard MIB structure for management agents, in analogy with the MIB-II standard for Internet MIBs.

From the five functional areas covered by the OSI management model, we have chosen to investigate in depth a configuration, fault and performance management architecture for Xunet (the remaining functional areas being security and accounting management). The fault management architecture enables us to monitor closely all the network switches for hardware faults, such as link level errors, buffer overflows, etc. The objective of the performance management architecture is to maintain and possibly improve the Quality of Service observed by the user at the call level (call setup time, call blocking probability) based on measurements extracted from the network control software. The manager is then able to affect performance of the entire network by controlling the distribution and allocated networking capacity of Virtual Paths (VPs) in the network. Network performance is also observed at the cell level, by computing the bandwidth utilization of virtual circuits and other useful cell-level statistics.

## 2.3 Implementation

As the basis of our OSI Management system, we have selected the OSIMIS software [KNI91]. OSIMIS provides an agent that manages the OSI transport layer of a Unix workstation, but can also pro-

vide support for more generic management applications. The package also includes a few management applications that retrieve information from the agents using the CMIP protocol. Our implementation expanded the agent with managed objects for Xunet and the management applications to include powerful graphics that depict the operational state of the network and a control interface that facilitates the overall management task.

The management applications typically run at Columbia University, or the AT&T Murray Hill site. TCP/IP sockets are used at the transport layer to connect to the agent at each site. Inside the agents, communication between logical and physical objects is achieved by using a proprietary protocol between the agent and the Xunet switch. For this purpose, we use UDP/IP packets travelling over a local Ethernet. The structure of the system is shown in Figure 2.

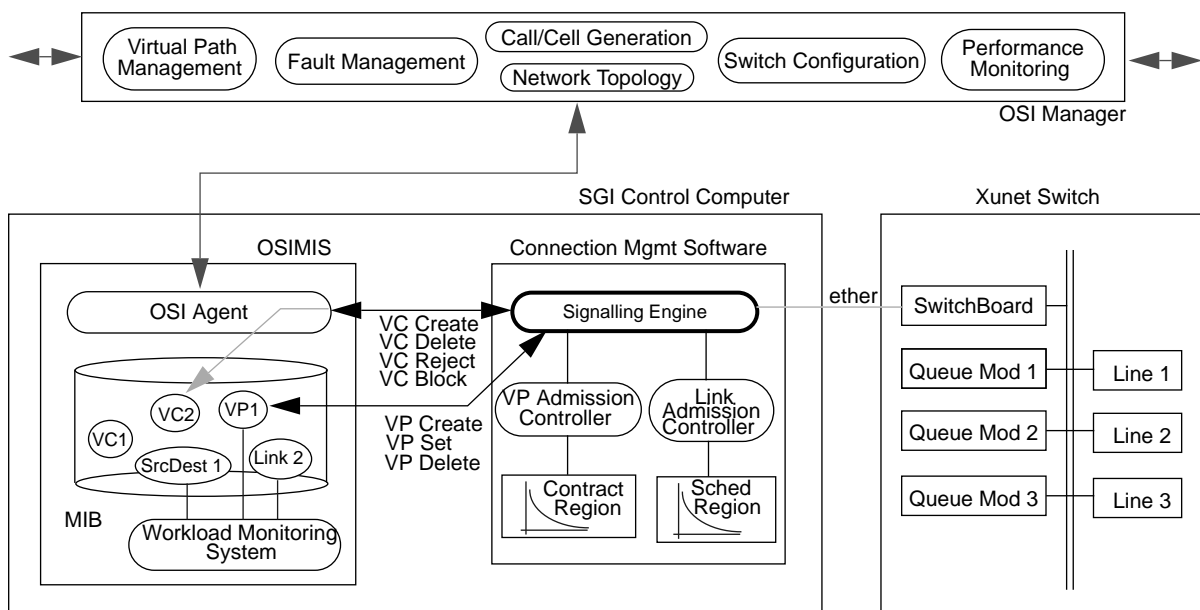


Figure 2: Structure of the Xunet Management System

## 2.4 The OSI Agent

The OSI agent contains managed objects for configuration, fault and performance management. The agent consists logically of two major groups of Managed Objects.

### 2.4.1 Hardware Configuration and Fault Management Group (HCFMG)

For the purpose of configuration and fault management, we have implemented managed object classes for each Xunet hardware module, such as *SwitchBoard*, *QueueModule*, etc. Each module is polled at regular time intervals by the agent to detect possible faults. A hardware fault triggers a notification inside the agent, which in turn can generate a CMIS Event Report primitive if the appropriate Event Forwarding Discriminator object has been created by the manager [ISO91b]. Currently, more than 300 different hardware errors can produce an equal number of event reports, something which gives extensive fault monitoring capabilities to the manager. The configuration and state of

the hardware modules is obtained from the Xunet switch every 20 seconds. The information is processed internally to update the corresponding managed objects.

The set of the hardware managed objects also gives complete configuration information of every switch. The management applications can display graphically the configuration and indicate the location of every generated event report.

#### **2.4.2 Performance Management Group (PMG)**

The PMG consists of a set of managed objects that closely monitor the performance of Xunet both at the cell and at the call level. In our model, one agent is installed at every ATM switch. The agent processes the information about call attempts appearing at the local switch. For each successful call attempt, an object of class *VCEntity* is created that models the corresponding Virtual Circuit connection (VC). The information is also processed by a set of objects dedicated to performance management at the call level, that measure quantities such as call arrival rates, call blocking probability, call setup time, call holding time, etc. [ISO92]. Call level performance is measured for three different abstractions that jointly give a complete view of network performance. In the first abstraction, the call-level performance is measured for outgoing links from the local switch. Only call requests that have selected this link are used to compute the call level performance statistics. These are stored in an object of class *Link*. There is one such object instance for each outgoing link from the local switch. Similarly, we measure the call-level performance for each Virtual Path that begins at the local cross connect, by computing the performance properties only for calls that use this VP to reach their destination. The information is stored in an object of class *VirtualPath*. Finally, if the call request originates from a network interface directly connected to the local-cross connect, an object of class *SourceDestination* (SD) is used to monitor the performance related properties between two nodes, regardless of the path that calls follow to their destination. SD objects are very useful to the manager for isolating two nodes and observing the call activity between these nodes only. This information is essential for making decisions on the VP distribution policy. In addition to measuring the call-level performance, Link and VirtualPath objects contain general configuration information for the corresponding links and VPs that originate from the current node.

As shown in Figure 2, the OSI agent receives four types of events from the local signalling entity: VC-Create, VC-Delete, VC-Block (when no network resources for the VC are available) and VC-Reject (any other cause), with the appropriate parameters. When a Managed Object of class *VCEntity* is created inside the MIB (upon a VC-Create event), it is logically related to the appropriate *Link*, *SourceDestination* or *VirtualPath* objects. Every 30 seconds, the Xunet switch is scanned to compute the number of cells transmitted or dropped for each VC. At the same time we use this information to update the total number of cells transmitted or lost on a Link, SourceDestination or VirtualPath based on the relations defined when the VC was created. In this way we can obtain also aggregate cell level statistics. The VC object is removed by a deletion event.

All 4 event types cause the agent to update some internal counters in the corresponding Link/SourceDestination/VP objects. Additional processing is performed at a regular time interval (usually set to 60 seconds). At that time, the agent calculates the values for the call-level performance related attributes, taking into account only the events that occurred during the past interval. For example, when a VC is created, a counter that registers the call arrivals is incremented. At the end of the 60

second period, the arrival rate is calculated, and the counter is reset to zero. All other attributes are calculated in a similar fashion.

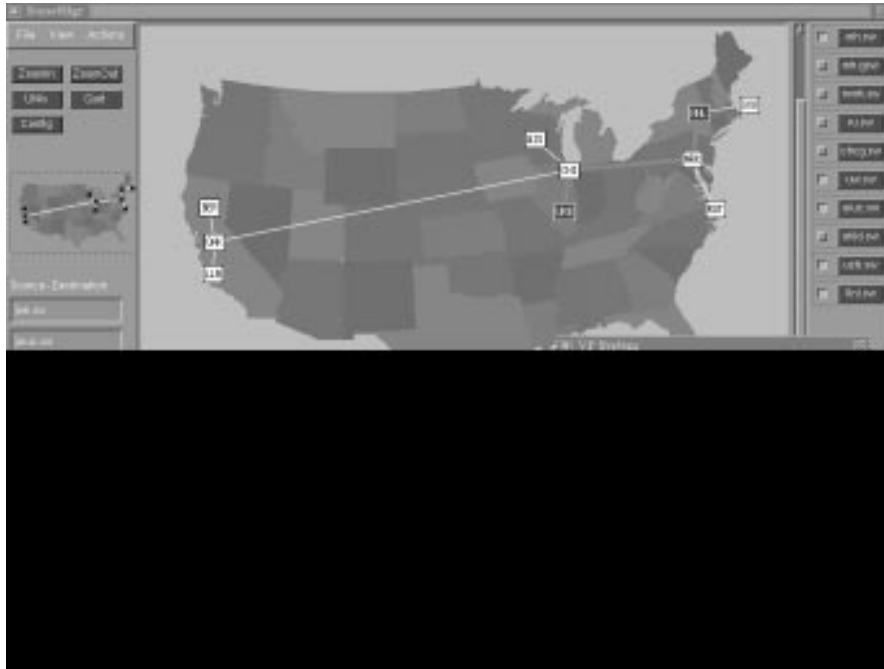
VP management functions are performed by the manager who communicates with the agents through the CMIP interface. When the management application issues an M-Create command with the appropriate parameters, a VP managed object inside the MIB is instantiated, and the Xunet signalling entity is informed to initiate a VP setup procedure. VPs can be subsequently modified by the M-Set command operating on the appropriate object, and deleted with an M-Delete command.

## **2.5 The OSI Manager**

Xunet is currently monitored and controlled through a Motif/X-Toolkit-based application. The same application is used to run controlled call and cell generation experiments. It consists of six tightly interconnected subsystems (Figure 2). Every subsystem contains the appropriate display tools and management interface functions for a specific management task:

1. **Switch Configuration:** Displays the hardware configuration of the switch using information from the objects of the HCFMG.
2. **Fault Management:** Receives OSI Event reports from the agents, that are related to hardware problems, and uses the Switch Configuration subsystem's display functions to inform the manager about the nature and location of the problem.
3. **Network Topology:** Displays a map of the network, with all switches, links and attached user-network interfaces (also called "Xunet Routers"). The displayed objects can be selected and observed independently. Support is also provided for displaying the route and configuration information of VPs.
4. **Virtual Path Management:** The manager is able to create and subsequently control Virtual Paths with M-Create and M-Set operations. The VP control task is guided by the observations obtained from the Performance Monitoring system.
5. **Performance Monitoring:** Collects the information that is provided by the PMG objects in each node and displays it using the functions of the Network Topology subsystem. The information can be either displayed in textual form, or graphically. In the latter case, we use time series plots that are updated in real-time. The plots allow us to observe the performance "history" of the network and the effects of VP management controls.
6. **Call and Cell Generation:** The Xunet signalling entities contain a call generation facility. A managed object inside the local agent makes it possible to control the call generation parameters in terms of destination nodes, call arrival rate and call holding time on a per traffic class basis. The call generation system can also be linked to the Xunet cell generator (to be described in the following section) for real-time cell generation.

The management application is located at Columbia University and serves as a comprehensive monitoring and control tool for Xunet. Figure 3 shows a capture of the application displaying the call level performance of the network.



**Figure 3:** The Xunet Management Tool

### 3. Infrastructure for Cell Level QOS Monitoring and Control

---

#### 3.1 The Need for a Cell Generation System

In order to validate and quantify the performance of the network, it is necessary to be able to provide different traffic loads. For example, in order to test the performance of different scheduling algorithms, loads close to the network capacity are necessary. In experimenting with such algorithms, a network developer cannot rely on real traffic loads provided by applications running on the network; not only are these applications independently operated and hence difficult to coordinate, but they are unreliable and, most importantly, not reproducible. Therefore, to avoid these problems we need to generate artificial traffic loads that emulate real application traffic.

Clearly, the effectiveness of this approach depends on the ability to accurately match the characteristics of the traffic to be emulated. In general, artificial traffic is generated according to traffic models that specify parameters such as the interarrival time between successive packets and the size of packets, according to a probability distribution. Typical examples include the Poisson model, commonly assumed for data traffic and the on-off model used for packetized voice traffic [SRI86]. Traffic produced from emerging applications such as video-conferencing, still image servers and real-time video transmission exposes more complex characteristics ([LAZ94], [MEL94]).

Therefore, building a network traffic generator requires meeting two conflicting requirements: implementing traffic models that accurately match real traffic characteristics, while achieving high traffic generation speeds in real-time. The former requires high computational complexity while the later calls for computational simplicity.