Managing VPs on Xunet III: Architecture, Experimental Platform and Performance

Nikos G. Aneroussis and Aurel A. Lazar

Department of Electrical Engineering and
Center for Telecommunications Research
Rm. 801 Schapiro Research Bldg.
Columbia University, New York, NY 10027-6699
e-mail: {nikos, aurel}@ctr.columbia.edu
Tel: (212) 854-2399

Abstract

An architecture for integrating the Virtual Path service into the network management architecture of future broadband networks is presented. Complete definitions of Managed Object Classes and behavioral descriptions are given. An experimental platform on top of the XUNET III ATM network provides the proof of concept. The Xunet manager is equipped with the necessary monitoring tools for evaluating the performance of the network and controls for changing the parameters of the VP connection services. Performance data from Xunet is presented to highlight the issues underlying the fundamentals of the operation of the VP management model such as the trade-off between throughput and call processing requirements.

Keywords: ATM, Quality of Service, Virtual Path Management, Performance Management

1. Introduction

Central to the operation of large scale ATM networks will be the configuration of the Virtual Path (VP) connection services. VPs in ATM networks provide substantial speedup during the connection establishment phase at the expense of bandwidth loss due to reservation of network resources. Thus, VPs can be used to tune the fundamental trade-off between call throughput and overall performance of the signalling system. They can also be used to provide dedicated connection services to customers such as Virtual Private Networks (VPNs). This important role of VPs brings forward the need for a comprehensive management architecture that allows the configuration of VP connection services by the network manager and the evaluation of the resulting network performance. Furthermore, call-level performance management will be essential to the operation of large ATM networks for making routing decisions and for long term capacity planning.

The review of the management efforts for ATM broadband networks reveals that there has been little work regarding the management of network services. In [OHT93], an OSI-based management system for testing ATM Virtual Paths is presented. The system is used exclusively for testing the cell-level performance of Virtual Paths, and allows the control of cell generators and the retrieval of information from monitoring sensors. The system is designed for testing purposes only and does not

have the capability to install Virtual Paths, regulate their networking capacity, or measure any call-level statistics.

A more complete effort for standardizing the Management Information Base for ATM LANs that meets the ATM Forum specifications is currently under way in the Internet Engineering Task Force (IETF) [IET94]. This effort focuses on a complete MIB specification in terms of configuration management, including VP configuration management. Performance management is also considered but at the cell level only. This modelling effort is intended to use the SNMP standard supported by the current SNMP agents managing the TCP/IP layers.

The ICM RACE project [ICM93] is defining measures of performance for ATM networks both at the call and at the cell level and the requirements for Virtual Path connection management. It is expected to deliver a set of definitions of managed objects for VP management and demonstrate an implementation of the architecture.

In [ANE93] we have described a network management system for managing (mostly monitoring) low level information on XUNET III. Our focus in this paper is on managing services, in particular, services provided by the connection management architecture. In order to do so, there is a need to develop an understanding of the architecture that provides these services. The integration of the service and management architectures can highly benefit from an overall network architecture model [LAZ93].

Within the context of a reference model for network architectures that we have previously published, we present an architectural model for VP connection setup under quality of service constraints. The architecture is integrated with the OSI management model. Integration here means that VPs set up by the connection management system can be instrumented for performance management purposes. The reader will quickly recognize that this instrumentation is representative for a large class of management problems such as billing (accounting), configuration management, etc.

We emphasized the following capabilities: monitoring Virtual Circuits (VCs) independently; monitoring and control of Virtual Paths (VPs); monitoring the call-level performance by computing statistics such as call arrival rates, call blocking rates, call setup times, etc.; control the call-level performance through allocation of network resources to Virtual Paths, and control of other operating parameters of the signalling system that influence the call-level performance, such as retransmission time-outs, call setup time-outs, call-level flow control, etc.

We have tested our overall management system on the Xunet ATM broadband network that covers the continental US. Finally, we have taken measurements that reveal the fundamental trade-off between the throughput and the signalling processing load as well as other quantities of interest that characterize the behavior of broadband networks.

This paper is organized as follows. Section 2 presents the architectural framework for managing VP connection services. Section 3 describes the Xunet III experimental platform and the implementation details of the VP management system. Network experiments with the objective of evaluating the management model and the performance of the network under several configurations of the VP connection services are presented in Section 4. Finally, Section 5 summarizes our findings and presents the directions of our future work.

2. Architecture

In this section we present an overall architectural framework for managing the performance of VP services on broadband networks. Underlying our modeling framework is the Integrated Reference Model (IRM) described in Section 2.1. The VP architecture embedded within the IRM is discussed in Section 2.2. The management architecture is outlined in section 2.3. Finally, in section 2.4 the integration of the service and management architectures is presented.

2.1 The Integrated Reference Model

To overcome the complexity problems in emerging broadband networks - caused by the variety of communication services to be provided, the required quality of service guarantees, the large number of network nodes, etc. - there is an urgent need for integrating management and real-time control tasks into a consistent framework. To this end, we have developed an overall model for network architectures called the *Integrated Reference Model (IRM)* [LAZ92]. In this model, the key role for network integration is played by the network telebase, a distributed data repository that is shared among network mechanisms.

The IRM incorporates monitoring, control, communication, and abstraction primitives that are organized into the *Traffic Control Architecture* and the *Management Architecture*, the *Information Transport Architecture* and the *Telebase Architecture*, respectively. The subdivision of the IRM into the Management and the Traffic Control Architectures on the one hand, and the Information Transport Architecture on the other, is based on the principle of separation between communications and controls. The separation between the Management and the Traffic Control Architecture is primarily due to the different time-scales on which these architectures operate.

The Integrated Reference Model is organized into five planes that model the above architectures (Figure 1). The Management Architecture resides in the network management or N- plane, and covers the functional areas of network management, namely, configuration, performance, fault, accounting and security management. Manager and agents, its basic functional components, interact with each other according to the client-server paradigm. The Traffic Control Architecture consists of the resource control, or M-, and the connection management and control, or C-, planes. The M-plane comprises the entities and mechanisms responsible for resource control, such as cell scheduling, call admission, and call routing; the C-plane those for connection management and control. The Information Transport Architecture is located in the user transport or U-plane, and models the protocols, and entities for the transport of user information. Finally, the Telebase Architecture resides within the Data Abstraction and Management or D-plane, and implements the principles of data sharing for network monitoring and control primitives, the functional building blocks of the N-, M-, and C-plane mechanisms.

The mechanisms within both the management and control architectures are modeled based on the classical monitoring/control paradigm. This means that a mechanism performs an operation by monitoring the network state, and, based on that knowledge, executes a control operation, leading to a change of that state. Therefore, sharing information among network mechanisms includes sharing of state data [LAZ93].

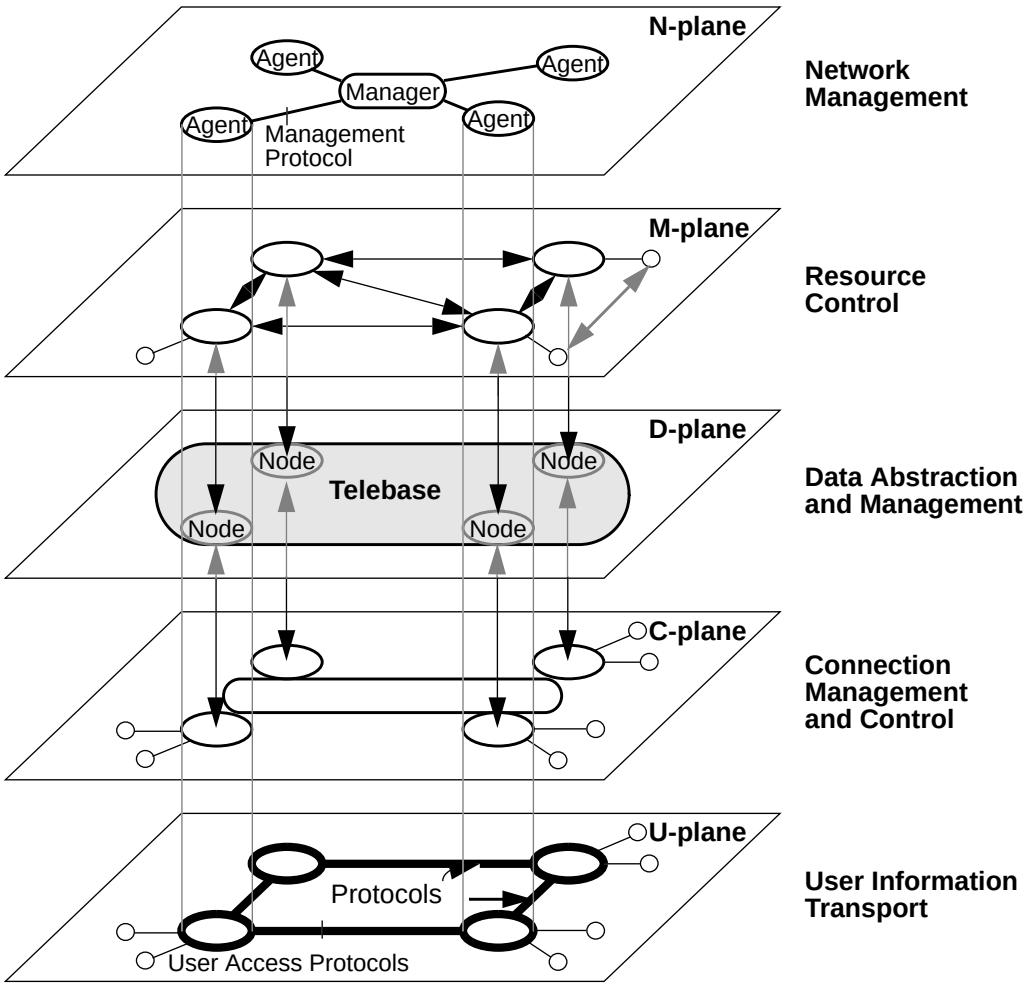

Figure 1: The Integrated Reference Model.

2.2 VP Architecture

The VP architecture closely follows the organization proposed by the IRM. It can be divided in two parts: the first part describes a model for establishing VPs, and the second a model for the integration of VPs with the signalling system. In either case, central to the VP architecture is the D-plane of the IRM. The D-plane contains all information regarding the configuration and operational state of VPs and is used by the algorithms of the other planes both for monitoring and control operations.

The establishment of Virtual Paths is performed by the signalling system. The latter resides in the C-plane. A signalling protocol is used to establish a Virtual Path hop by hop. At every node along the route of the VP, the necessary networking capacity must be secured from the outgoing link that the VP is traversing. The networking capacity of links is described by the Schedulable Region (SR) [HYM91] and of Virtual Paths by the Contract Region (CR) [HYM93b]. Informally, the Schedulable Region is a surface in a k dimensional space (where k is the number of traffic classes), that de-

scribes the allowable combinations of calls from each traffic class that can be accepted on the link and be guaranteed Quality of Service. An analogous definition is given for the Contract Region (CR) of Virtual Paths. The Contract Region is a subregion of the SR reserved for exclusive use by the VP. If the requested capacity allocation of a VP cannot be achieved, the allocated capacity at the end of the VP establishment phase is the minimum capacity available on the corresponding links.

Each Virtual Path can accommodate calls belonging to different service classes. Usually, it is necessary for each call to provide some appropriate parameters to the link multiplexer describing the service class that it belongs to. This must be done at all cross-connect points (nodes) along the path during the call establishment phase. This has the following implication when VPs are in use: each VP must partition its space of Virtual Circuit Identifiers (VCIs) among traffic classes during the installation of the VP, and for each such circuit group, set the appropriate parameters on each link multiplexer according to the service class that it belongs to. The set of all Virtual Paths in the network, characterized by their route, Contract Region and other configuration information, including the partitioning of the VCI space among traffic classes, comprise the *VP distribution policy*. The VP distribution policy is stored in the D-plane.

At the source node of every VP operates an admission controller. The admission controller is an entity in the M-Plane that formulates the admission control policy by defining the Admissible Load Region (ALR) for the VP [HYM93a]. The ALR provides a measure for the capacity of a link multiplexer on the call level. The ALR is used by the signalling algorithm of the C-plane to make admission control decisions for incoming call requests. Thus, the VP architecture represents a connection service installed in the D-plane. The policy to accept or deny user access to the service is also encoded in the D-plane.

Figure 2 shows the interaction between entities in the various planes of the IRM that provide the VP

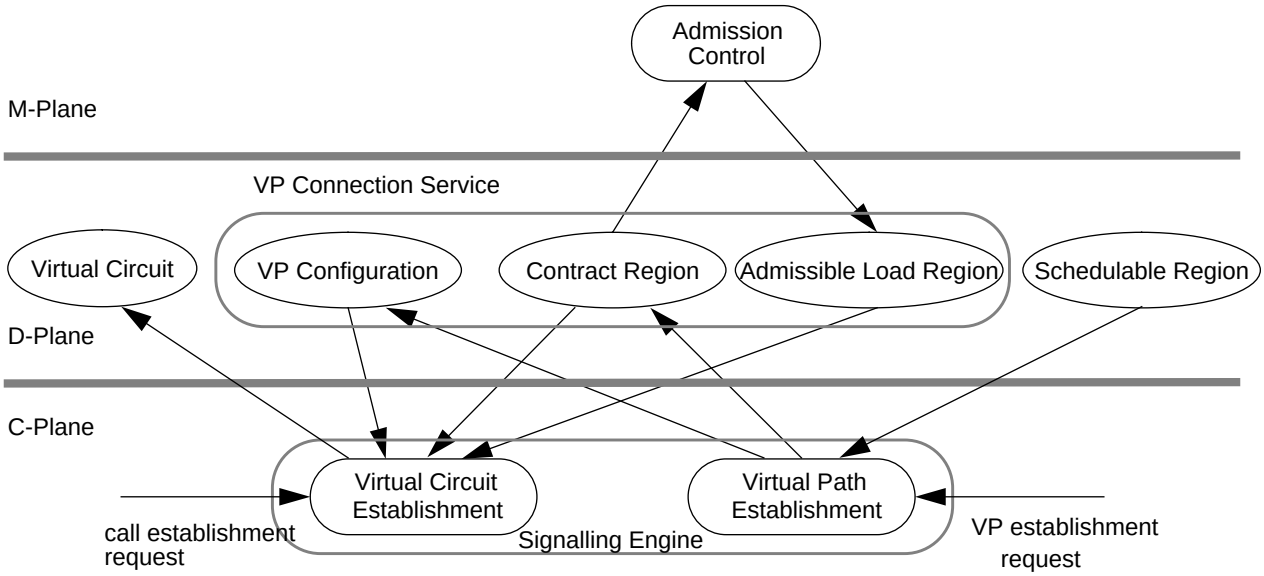

Figure 2: Flow of Information during Installation and Operation of the VP Connection Service

connection service. During the VP establishment phase, the signalling engine receives the request for the VP to be setup and requires the link controllers of the outgoing links to verify whether the

necessary networking capacity is available. In order to demonstrate in a simple manner the principles of operation of our model, we deliberately omitted any interactions with other objects such as routing tables, etc.

After the VP was established, it is represented in the D-plane by a logical group of three objects: The VP configuration object contains general VP configuration information such as the input and output port numbers, the allocation of the VCI space, the VP operational state etc. The object is subsequently read by the signalling engine during call setup requests attempting to use the VP. The same applies to the Contract Region object, which describes the capacity of the Virtual Path, and the operating state of the VP admission controller as a point in the CR. The ALR object contains the admission control policy and is only written by the Admission Controller, after examining the Contract Region object.

During the VC establishment phase, the signalling engine reads the VP configuration object to determine if the VP can be used to reach the desired destination. It also reads the CR and ALR objects to examine if the call can be admitted on the VP. When the call has been established, a Virtual Circuit object is created in the D-plane that contains all necessary information for the VC. This information includes VC configuration such as the VP Identifier (VPI) and VC Identifier (VCI), the traffic descriptor used to allocate resources, and other parameters for performance monitoring such as the incoming and outgoing cell rates, etc.

Virtual Paths can be used in two ways. If the VP is terminated at the Customer Premises Equipment (CPE), the customer is controlling the VP admission controller. In this case the VP can be regarded as a dedicated virtual link (or pipe) of a rated networking capacity and circuit capacity. A network composed of such VPs terminated at the customer premises is also known as a Virtual Private Network (VPN). The Network Manager has the capability to configure and maintain a VPN by managing the individual VP components according to the customer requirements.

Alternatively, the termination of VPs may not be visible to the network customer. In this case, VPs are used by the network operator to improve the performance of the signalling system, the availability of resources between a pair of nodes, or even improve certain call level measures of Quality of Service for the customer, such as call setup time and blocking probability.

2.3 Management Architecture

The Management Architecture builds on the OSI Management 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, which 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.

Management operations take place in the N-plane of the IRM (Figure 1). The MIB of every agent is located in the D-plane of the IRM. As a result, the linking of the logical objects in the MIB with real objects is done within the D-plane [TSU92]. Control operations from the manager applied to objects in the MIB are reflected in the state of the real objects of the D-plane, which in turn, affect the behavior of the algorithms in the C- and M- planes. Conversely, the change of state of the real objects in the D-plane will cause an update of the state of the logical objects in the MIB.

Therefore, in our model, monitoring and control of the VP architecture is possible by defining the appropriate managed objects in the MIB and linking them to the appropriate entities of the D-plane. What managed objects to define, how to integrate them in the D-plane and how to define their behavior will be the topic of the following section.

2.4 Integration of the VP and Management Architecture

The purpose of this section is to describe the object level model for VP management and its integration within the D-plane. Management of VPs takes place in the N-plane. The network manager decides on a VP distribution policy and implements this policy by issuing commands to the agents installed across the network. The manager is able to install a Virtual Path, remove it, or change its networking capacity and admission control policy. Although the capabilities to install and control VPs are essential for the management system to implement a VP distribution policy, it is also essential for the manager to evaluate the performance of the network under a given VP distribution policy. For this reason, a generic performance management model (in addition to the VP management model) both at the call and the cell level becomes necessary. Note however that VP management operations stem from the call-level performance management model, and therefore, the VP management model can be considered as part of the latter.

The performance management model consists of a set of *quantities* that measure network performance, and a set of *controls* that affect this performance. A set of *rules* use the performance measures to derive the necessary controls that will reach a *performance objective*. Appendix A presents the quantities that measure performance both at the call or the cell level. These quantities together with a set of controls must appear in the definition of Managed Object Classes (MOCs) for performance management.

In our model, one agent is installed at every ATM switch cross connect (also referred to as the “local cross-connect”). The agent processes the information about call attempts from the incoming links. For each successful call attempt, an object of class *VcEntity* is created for the corresponding Virtual Circuit connection (VC). The object contains the necessary attributes for cell-level performance monitoring, as given in Appendix A. At the same time, the information about call attempts is processed by a set of objects dedicated to performance management at the call level (see Figure 3).

Call level performance can be measured for three different abstractions to give a complete view of network performance. In the first abstraction, the call-level performance (as defined in Appendix A) is measured for outgoing links from the local cross-connect. Only call requests that have selected this link are used to compute the call level performance statistics. These are stored in an object of

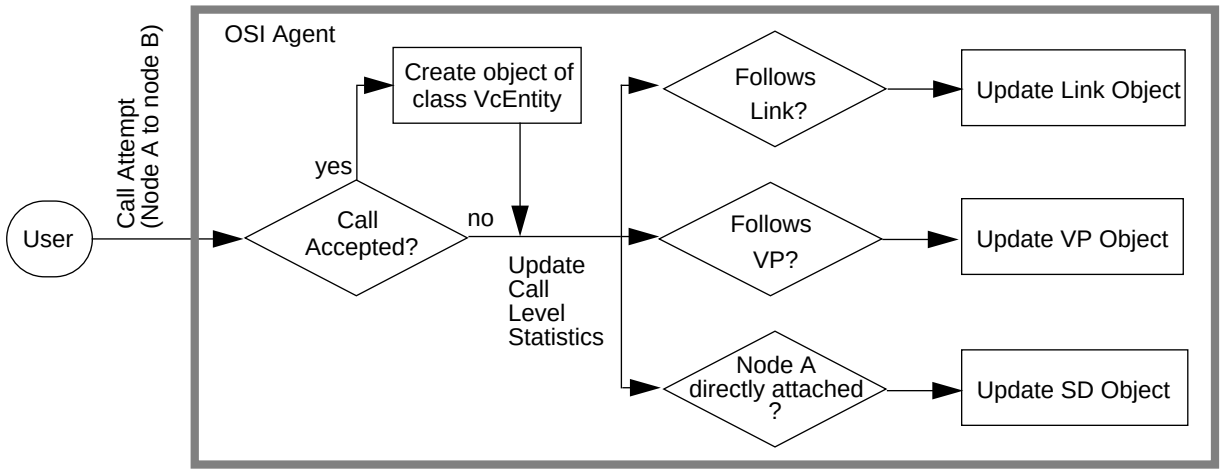

Figure 3: OSI Agent Functional Diagram

class *Link*. There is one such object instance for each outgoing link from the local cross-connect. 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, as we will see in a following section, for making decisions on the VP distribution policy.

As discussed in the previous section, the MIB of every agent resides in the D-plane. Managed Objects use the information stored in the D-plane to update their state. Figure 4 demonstrates this principle: Managed Objects of class *VcEntity* represent the Virtual Circuit object that was created in the

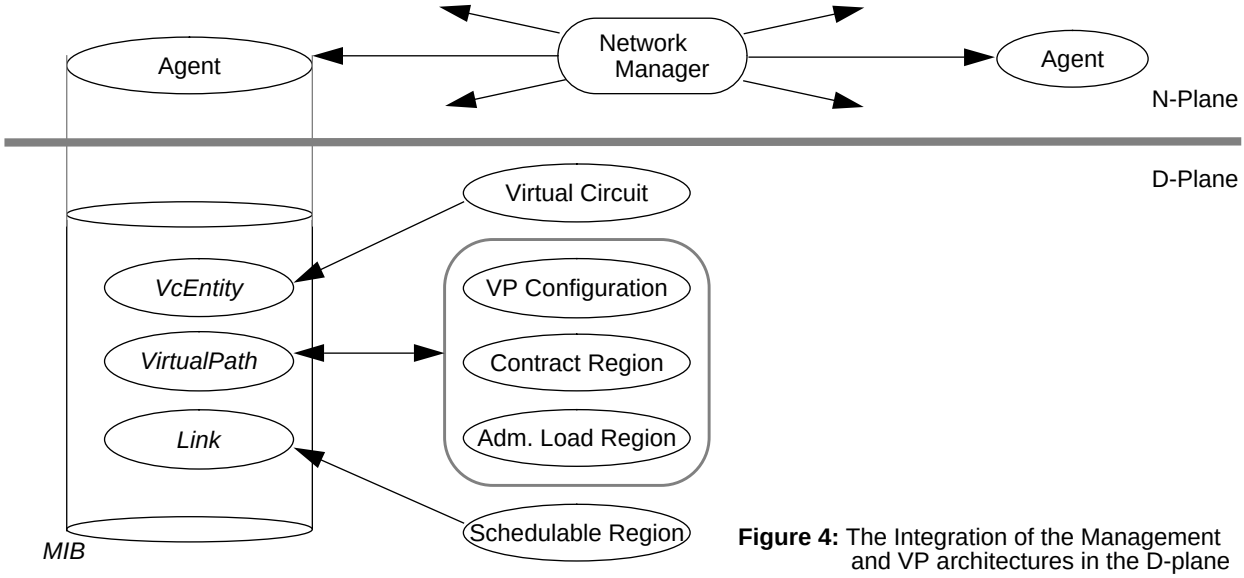

Figure 4: The Integration of the Management and VP architectures in the D-plane

D-plane by the signalling system. The attributes of the managed object mirror the state of the corresponding real object. In the same manner, the MO of class *VirtualPath* contains attributes that reflect the state of the corresponding real objects (VP Configuration, Contract Region and Admissible Load Region). An MO of class *Link*, uses the object Schedulable Region (among other information), to reflect the state of the link Schedulable Region on one of its attributes. Additional processing of events (such as VC creation, etc.) inside the agent can provide the necessary call-level performance related properties (such as call arrival rates). These might not be readily available from other objects of the D-plane.

A detailed description for the MOCs related to cell level performance management (*VcEntity*), and call-level performance management (*Link, VirtualPath, SourceDestination*), and also for other MOCs that are used in our model follows. Figure 5 shows the inheritance tree we are using. Some MOCs are specific to Xunet hardware and will be described later in this paper.

The class *VcEntity*, directly derived from *Top*, models each Virtual Circuit connection. Each VC object contains configuration information such as the number of the input and output slot, Virtual Path Identifier (VPI) and Virtual Circuit Identifier (VCI). In ATM terminology, this implies that the VC object models the input and output Virtual Circuit Link (VCL) at the local cross-connect. Thus, the end-to-end management of a VC that spans many cross-connect points (and hence has one instance in each OSI agent at every cross-connect) is achieved by managing the individual objects in combination [MAZ91]. Additional attributes for each VC include the call establishment time, traffic descriptor (composed of a service class characterization and the allocated networking capacity in kilobits per second), adaptation layer information and source/destination end-user service information. The package *cellPerformancePackage* contains attributes associated with the cell-level performance related parameters, and will be described below.

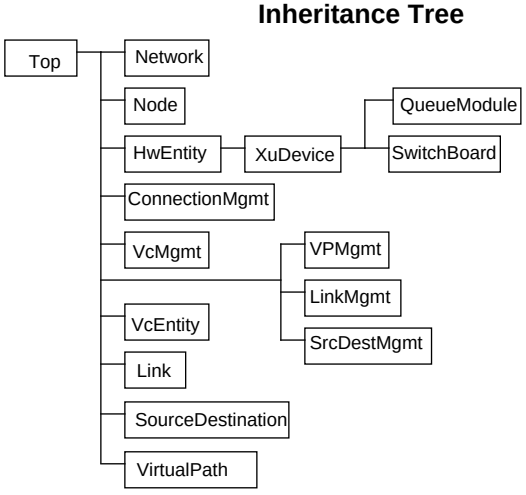

Figure 5: The Inheritance Tree

The class *VirtualPath* derived from *Top* is used to describe a Virtual Path (VP). The VP object in analogy with the VC object is comprised of an incoming and an outgoing part at each cross-connect point. At the VP source or termination point, the VP has only an outgoing / incoming part respectively. Attributes used to describe the configuration of the Virtual Path are: *vpIdentifier* (VPI), *vp-*

Source, *vpDestination* (VP source and termination address), *circuitCapacity* and *timeEstablished*. The VP object at the source also contains a *callPerformancePackage*, and an *admissionController-Package.* These will be described below.

The class *Link* is derived from *Top* and is used to model input or output network links. The mandatory attributes of this class are *linkType* (incoming or outgoing), *linkModuleDescription* (describes the hardware of the link interface), *linkSource, linkDestination* and *linkSlotNumber* (the slot umber the link is attached to). If the link is outgoing, it also contains a *callPerformancePack*age, and an *admissionControllerPackage*.

The class *SourceDestination* is used to describe the call level activity between a source node and a destination node. A Source-Destination (SD) object exists in the agent if there is call-level activity between the two nodes (i.e., call attempts originating at the source node, travelling through the local cross connect and terminating at the destination node) and the source node is either the local cross-connect, or a User-Network Interface (UNI), which is directly attached to the cross connect (i.e., attached at the other end of an incoming Link. The SD object contains the following attributes: *sourceNodeAddress* and *destinationNodeAddress* and a *callPerformancePackage*.

The *callPerformancePackage* is an optional package that measures the call-level performance as outlined in Appendix A. It is contained in all objects of class *SourceDestination*, and measures all call-level activity between the two endpoints. Objects of class *Link* contain the package if they are outgoing from the local cross-connect. In that case, the package measures the activity for calls that follow the link and do not belong to a virtual path that uses the same link. Finally, an object of class *VirtualPath* contains the package only if it is representing the source point of the virtual path. The package measures the activity of call requests that use the virtual path. The attributes of the *callPerformancePackage* are the following: *activeCircuits*, *callArrivalRate* (average arrival rate of requests in calls/min), *callArrivedCounter* (counter of call requests), *callResourceBlockedCounter* (counter of calls blocked due to resource unavailability), *callErrorBlockedCounter* (counter of calls blocked due to protocol errors, e.g., time-outs, etc.), *callBlockingRate* (average rate of calls blocked for any reason in calls/min), *setupTime* (average time to establish the connection in milliseconds), *holding-Time* (average duration of connections in seconds), *numExchangedMessages* (average number of messages that have been exchanged to setup the connections, as an indicator of the processing required for each connection), and *measureInterval* (the time in which the above averages are computed in seconds). All quantities are measured separately for each service class, and then a total over all classes is computed. The definition of a service (traffic) class (e.g., voice, video, etc.) that we are using can be found in [HYM91]. Thus, we separately evaluate average arrival rates, blocking rates, etc., for voice and video calls, and then compute the averages over all calls.

The *cellPerformancePackage* measures the cell-level performance. The attributes *cellTransmitted-Counter*, *cellTransmissionRat*e, *cellDroppedCounter* and *cellDroppedRate* measure the number of cells transmitted or blocked and their respective time averages. The attribute *avgCellDelay* measures the average time from the reception till the transmission of cells in the local cross-connect. The package is included in objects of class *VcEntity*, where the cells belonging only to the VC are measured. As an option, it can also be included in objects of class Link, SourceDestination or VirtualPath. In the latter case, a sum of the attributes over all VC objects that belong to the Link/SourceDestination/ VirtualPath is computed, and the respective attributes of the Link/SourceDestination/VirtualPath

objects are updated.

The package *admissionControllerPackage* is mandatory for outgoing links and virtual path objects starting at the local cross-connect. It describes the state of the admission controller, which is located at the output links (for switches with output buffering) and at all virtual path source points. The package contains the following attributes: *networkingCapacity* (the schedulable region for link objects or the contract region for virtual path objects), *admissionControllerOperatingPoint* (the operating point of the admission controller given the established calls for each traffic class), *admissionControlPolicy*, *admissionControllerOperationalState* (enabled (call requests allowed to go through and reserve bandwidth) or disabled) and *admissionControllerAdministrativeState*.

The class *ConnectionMgmt* contains attributes that control the operation of the local signalling entity. There is only one instance of this class in every agent. Attributes of this class are the following: *signallingProcessingLoad* (an index of the call processing load observed by the signalling processor), *maxSignallingProcessingLoad* (the maximum signalling load value allowed, beyond which the signalling processor denies all call establishment requests), *signallingRetransmitTimeout* (the timeout value in milliseconds for retransmitting a message if no reply has been received), and *signallingCallSetupTimeout* (the maximum acceptable setup time in milliseconds for a call establishment. If the time to establish a circuit is more than the current value, the circuit is forced to a tear-down.). The single instance of this class is also used to contain four other container objects of class *LinkMgmt*, *SourceDestinationMgmt*, *VirtualPathMgmt*, and *VirtualCircuitMgmt*. There is only one instance from each of these four classes, which is used to contain all objects of class *Link*, *SourceDestination*, *VirtualPath*, and *VirtualCircuit*, respectively.

The purpose of the above description is to give an overview of the managed object classes and attributes for performance management. For simplicity, we omitted the definition of associated thresholds for each performance variable that can trigger notifications in case of threshold crossing [ISO92]. Such definitions can be easily incorporated in the above model. Appendix B contains a GDMO [ISO91a] description of all Managed Object Classes related to performance management.

3. Experimental Platform

3.1 The Xunet ATM Testbed

Xunet is one of the five Gigabit testbeds (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 continental United States [FRA92]. Figure 5 shows the topology of the network and the configuration of the management system. The network links are currently rated at 45 Mbps and are gradually substituted with 622 Mbps links. Access at every node is provided by 200 Mbps network interfaces. A variety of standard interfaces (TAXI, HiPPI, etc.) is under development and will be available in the near future. A workstation serves as the switch Control Computer (CC) at each network node. The CC runs the switch control software that performs signalling, control and fault detection functions. Xunet is an ideal broadband networking platform for experimentation, not only for high-bandwidth networked applications, but also for the connection and resource management software that will dominate the emerging ATM networks with QOS

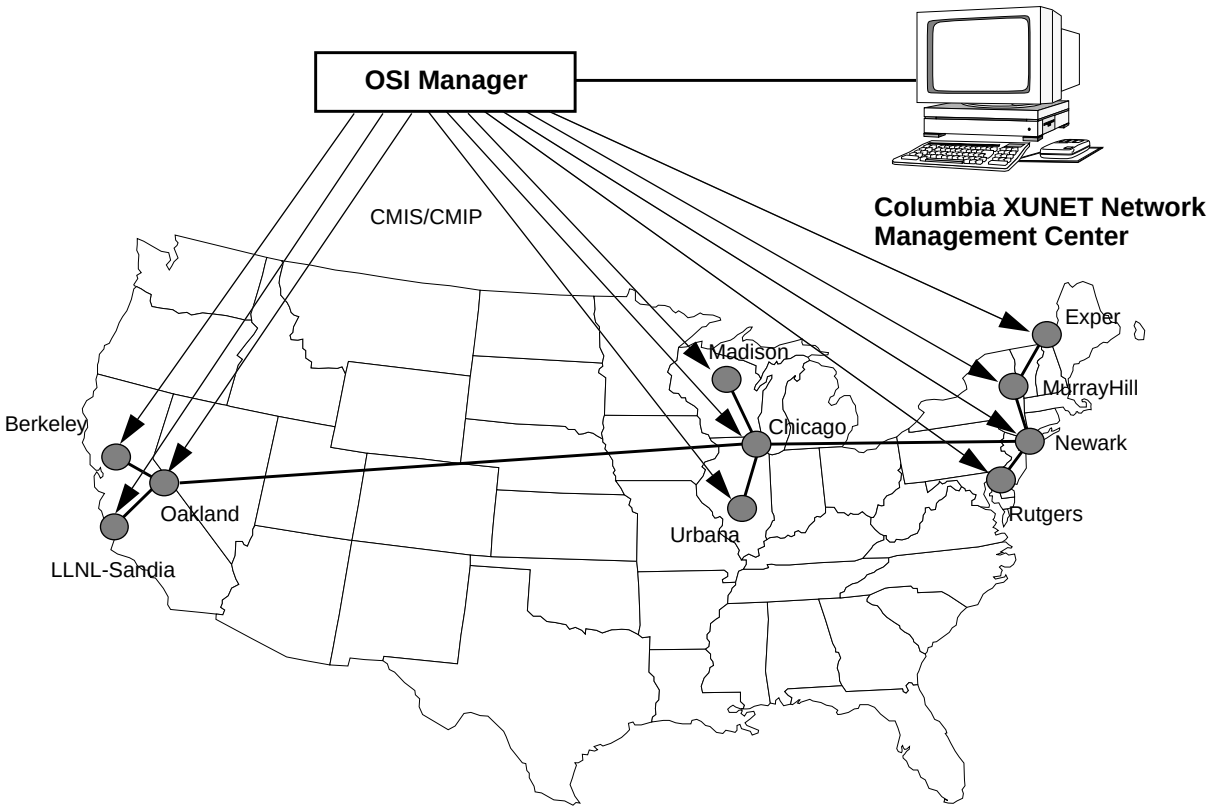

Figure 6: The Xunet Management System

guarantees.

3.2 The Xunet VP Signalling and Admission Control Architecture

Xunet supports five traffic classes. Class 1 is used for high priority control messages and is given absolute priority by the link scheduler (multiplexer). Class 2 is used for Video service, Class 3 for Voice, Class 4 for priority data and Class 5 for bulk data [SAR93].

A signalling system very similar in characteristics with the CCSS#7 (Common Channel Signalling System) has been installed on Xunet. The system allows virtual circuit establishment with best effort resource allocation in a single pass. An admission controller using the Schedulable Region operates on every outgoing link. Since Class 1 traffic has very low volume, it is omitted from the computation of the schedulable region. In addition, Class 4 and 5 traffic are given the remaining multiplexer capacity after resources for Class 2 and 3 traffic have been allocated. The Schedulable Region can be adjusted and the Admission Control policy dynamically modified from the management station. Currently, the admission control policy used is complete sharing [HYM93a].

Virtual Path establishment is also one pass with best effort resource allocation. When the VP has been established, an admission controller is activated at the source node of the VP and uses the allocated contract region for admission control decisions. The admission control policy is again complete sharing. A signalling channel is also established at both VP termination points to be used for VC establishment messages using the VP, in the same way as the signalling channel used on every

link. As a result, from the point of view of the signalling system, VPs are considered as regular links with only minor differences.

The Contract Region of VPs can be changed dynamically along all the nodes in the path. Deallocation of resources is straightforward. Resource allocation (when the Contract Region is to be expanded) is performed in the same way as in the VP establishment phase. Finally, when a VP is removed, the Contract Region is returned to the Schedulable Regions of the links along the path, all VCs using the VP are forced to termination and the VP signalling channel is destroyed.

Since the Xunet topology is a tree, the current routing tables are static. At each node, every incoming VC establishment request has three options: follow a specific VP to the next hop; follow any VP to the destination, or follow the usual VC setup procedure. In the first case, if the designated VP is full, the call is rejected. In the second case, if no appropriate VP is found, the call follows the normal call setup procedure by trying to allocate resources on one of the outgoing links. In the last case, the VC is not allowed to follow any VP.

3.3 The Xunet OSI Management System

From the five functional areas covered by the OSI management model, we have chosen to implement a configuration, fault and performance management architecture for Xunet (the remaining functional areas being security and accounting management). The configuration and fault management architecture enables us to monitor closely all the network switches for hardware faults, such as link level errors, buffer overflows, etc. The performance management architecture builds on the performance management model (managed object definitions and behavior) that was presented in the previous section.

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 provide 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 over a local Ethernet. The structure of the system is shown in Figure 7.

3.4 The OSI Agent

The OSI agent contains managed objects for configuration, fault and performance management. Figure 8 shows the containment tree for the MIB. The agent consists logically of two major groups of Managed Objects:

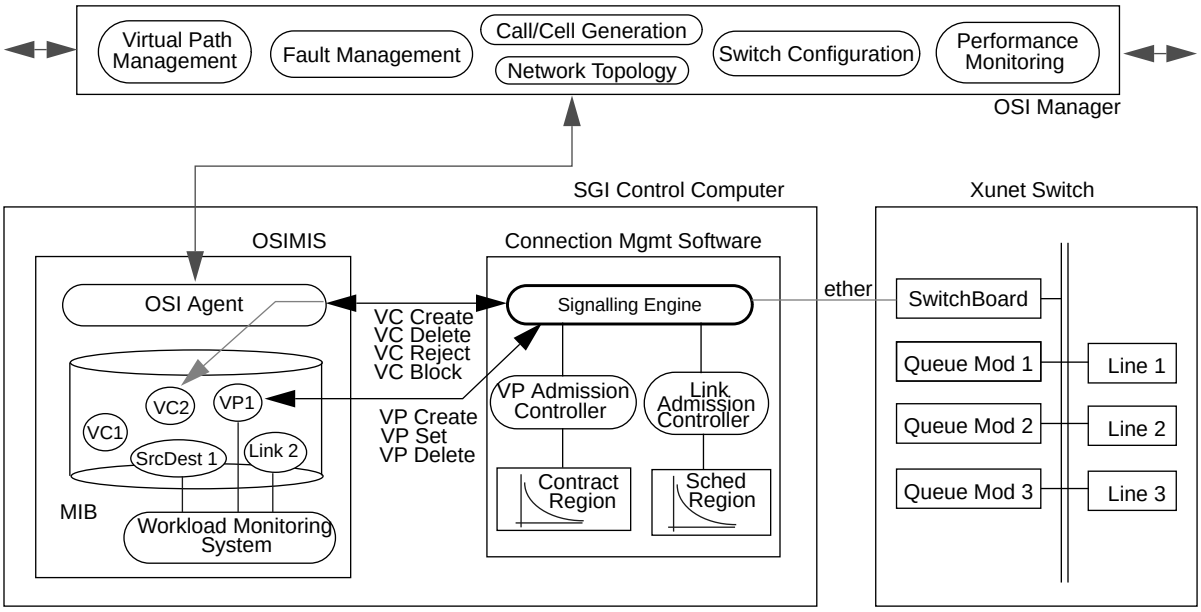

Figure 7: Structure of the Xunet Management System

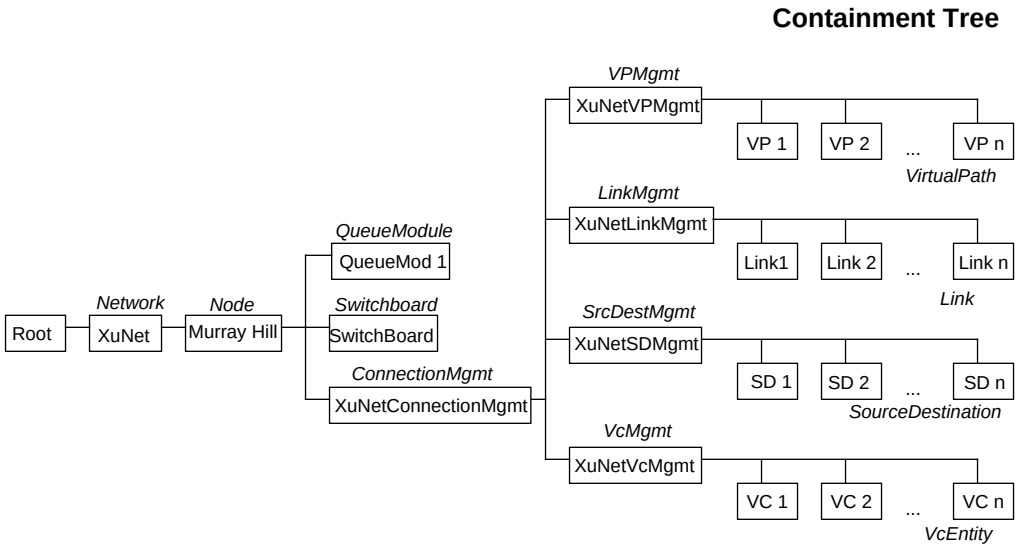

Figure 8: The Containment Tree of the Xunet MIB

3.4.1 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.

3.4.2 Performance Management Group (PMG)

The PMG consists of a set of managed objects that monitor closely the performance of Xunet both at the cell and at the call level. All call level information is obtained from the local signalling entity. The OSI agent receives four types of events: VC-Create, VC-Delete, VC-Blocking (no network resources) and VC-Rejection (any other cause), with the appropriate parameters. A proprietary messaging protocol is used for communication between the agent and the local signalling entity. Upon a VC creation event, a Managed Object of class *VcEntity* is created inside the MIB that contains all the available information on this VC. The object is 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. 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 (controllable by the manager through the *measureInterval* attribute, and usually set to 60 seconds). At that time, the agent calculates the values for the performance related attributes that were described in the previous section, 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.

Parameters of the signalling entity are controlled through M-Set operations on attributes of the *ConnectionMgmt* object. Each Set operation causes a control message to be sent from the agent to the signalling entity.

3.5 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 on Xunet. It consists of six tightly interconnected subsystems (Figure 7). 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 in each node and displays it using the display functions of the Network Topology subsystem. A history mechanism extracts history traces of observation frames that are used to produce time series plots. These plots, which are dynamically displayed and updated, are used to observe the long term performance of the network.
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 traffic generator 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 9 shows a capture of the application displaying the call level performance of the network.

4. Performance

We are currently using the management system to run controlled experiments on Xunet to study the call and cell level performance of the network. Cell-level experiments consist of loading the network with a mix of voice and video calls and measure quantities such as the cell loss rates, average cell delay, etc. These are used for evaluating the Schedulable Region of each link. Call level experiments consist of loading the signalling system with an artificial call load. A Call Generator on every network cross-connect produces requests for calls with exponentially distributed interarrival and holding times. There is no cell generation after calls have been established. The purpose of the call generation experiments is to evaluate the performance of the signalling system and of various VP distribution policies. In the rest of this section we will focus on the objective of performance management at the call level and will demonstrate results from various call-level experiments conducted on Xunet.

Figure 9: The Xunet Management Tool

4.1 Semantics of Performance Management

The purpose of performance management at the call level can be summarized in the following:

- Minimize call blocking due to unavailability of network resources. This unavailability can be caused by several factors including a faulty link, a poor VP distribution policy, a routing malfunction, an overloaded signalling entity, etc.
- Minimize the call setup time. The call setup time is perceived by the user as a measure of the quality of service offered by the network. High call set up times may prompt the user to hang-up leading in loss of revenue for the network.

In our context, the main goal of call level performance management is to evaluate the fundamental trade-off between the throughput resulting from allocating networking capacity to virtual paths and the load on signalling processors.

In our setting, the manager connects to many agents installed at every network switch, and retrieves the performance information stored in the appropriate managed objects. We have described three MOCs with performance related attributes: The *Link* and *VirtualPath* MOCs represent a concrete

network entity and measure the respective call request activity. The *SourceDestination* MOC is introduced to facilitate the manager's observations between two endpoints. If this MOC was not available, the manager would need to process information from many Link and VirtualPath objects to derive the desired statistics. With the introduction of the SourceDestination MOC, this task becomes significantly easier.

The main task of the manager is to collect measurements in regular time intervals, and evaluate the performance of the network. If the performance is not satisfactory (high blocking and high call setup times), the manager can apply the following controls:

1. Create a Virtual Path between two endpoints and allocate resources to it. This action alleviates the intermediate nodes from processing call requests and decreases the call setup time.
2. Delete a Virtual Path responsible for the non-satisfactory performance. This course of action may be taken because the maximum number of VP terminations has been reached and new VPs cannot be created in the system, or because there is no offered load to the VP, or because a new VP distribution policy has been decided and the VP topology must change.
3. Change the allocated networking capacity of a Virtual Path either by releasing a part of or increasing the allocated resources. This control is performed when the load offered to the VP has been reduced or increased.
4. Change signalling parameters, such as the time-out for call setups, the time-out for message retransmissions and the maximum allowed signalling load (which is a window-type control on the number of requests handled by the signalling processor). These parameters affect the call blocking rates, but also the average call setup time.

With the above in mind, the call-level experiments have been separated in two major phases. In the first phase, we measure the performance of the signalling system without using VPs. This experiment allows us to find suitable values for the parameters of the signalling entities that give the highest call throughput. The second phase builds upon the first phase, and attempts to determine the call throughput by measuring the performance of the network with VPs in place.

4.2 Performance of the Signalling System

In this phase, the network is loaded with an artificial call load. Our goal is to measure the performance of the signalling system under heavy loads without Virtual Paths. Call generators exist in every node, and can be controlled from the management station. Generation is controllable for each destination in terms of the call arrival rate and call holding time for each of the five traffic classes supported by Xunet. The network was homogeneously loaded from five sites (Murray Hill, Rutgers U., U. of Illinois, Berkeley and LLNL) with Poisson call arrivals and an exponential holding time with a mean of 3 minutes. We used only calls of Class 2 (assumed to consume a peak of 6 Mbps/call) and Class 3 (64 Mbps/call) in a ratio of 1:10 (i.e., the arrival rate of Class 3 calls is 10 times larger). All the links in the experiment are of 45 Mbps capacity. The schedulable region of each link is given by a two dimensional hyperplane. We used peak rate allocation and in this scheme, the SR can accommodate a maximum of 7 calls of Class 2 or 703 calls of Class 3. The admission control policy used is complete sharing.

Figure 10 shows the measurements obtained by gradually increasing the call generation rate. Each

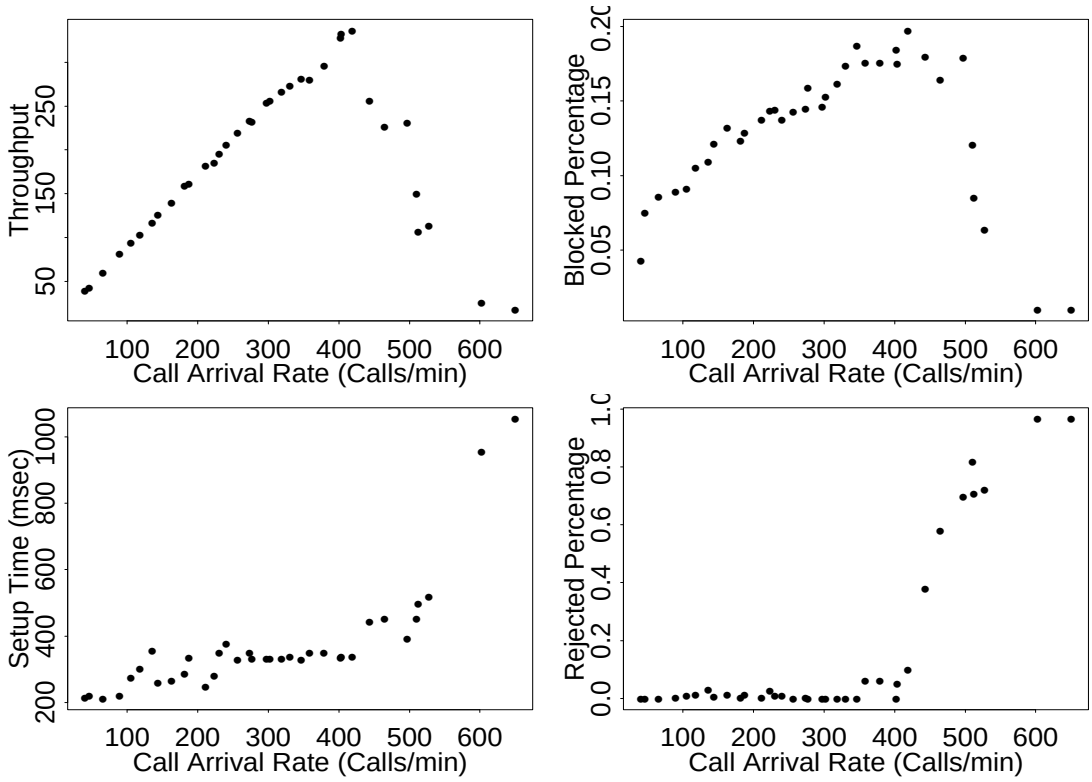

Figure 10: Performance of the Signalling System

measurement is computed by adding the respective measurements for Class 2 and Class 3 traffic. Both the call throughput and call blocking due to resource unavailability (the “Throughput” and “Blocked Percentage” curves) rise almost linearly with the call arrival rate. The sudden drop in the total call throughput is due to overloading the signalling system with call setup requests. At that point, the call setup time starts to rise rapidly, and the call blocking curve drops. This is due to the fact that, during overload, calls are rejected before they can contend for resources. The “Rejected Percentage” plot shows the percentage of calls that are rejected due to any other signalling error The curve is almost zero in the normal operating region of the network, and rises rapidly during overload.

The congestion situations seem to appear first at the Newark and Oakland switches, that are the first to become overloaded with call request messages. It is therefore essential for the network manager to regulate the call arrival rate at the entry points in the network. This can be done by setting an appropriate value for the *maxSignallingProcessingLoad* attribute of the ConnectionMgmt object. The signalling load is computed from the number of signalling messages received and transmitted from the switch in the unit of time. If the load reaches the *maxSignallingProcessingLoad* value, all incoming call requests are discarded. We have found experimentally that by restricting the signalling load to about 450 messages per minute at the nodes connected to the call generators, the network operates within the capacity of the signalling processors. However, the value of the maximum

throughput depends on the loading of the network, (in our case, we used homogeneous loading), and can vary between signalling processors as the generation parameters change.

4.3 Call Level Performance

The second phase has the objective of measuring the call-level performance under different VP distribution policies. In order to study the trade-off between the blocking probability and the allocation of networking capacity to VPs, we have performed the following experiment on the east coast segment of the network (the rest of the network was unavailable until July' 94). This four node segment consists of two nodes in MurrayHill NJ (called MHEX and MH), one node in Newark NJ (NWRK), and one at Rutgers University (RUTG) connected in tandem.

The generation ratio between Class 2 and Class 3 calls was 1:100. The call arrival rate from each call generator was kept fixed throughout the experiment. The generator at MHEX produces traffic to NWRK (180 calls/min) and RUTG (210 calls/min). The generator at MH produces only traffic to RUTG at 180 calls/min. The generator at RUTG produces traffic to NWRK at 180 calls/min. One VP is established from MHEX to RUTG (Figure 11). The signalling overload control that was de-

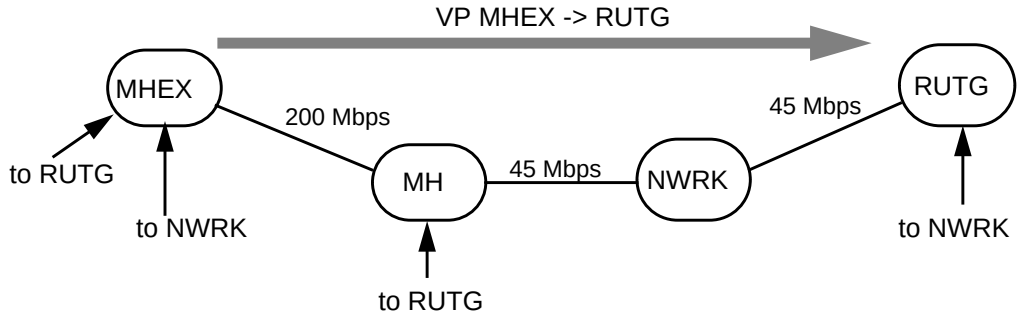

Figure 11: Network Topology for the VP experiment

scribed in the previous experiment was disabled.

Only the traffic from MHEX to RUTG is allowed to follow the VP. Calls that find the VP full and the calls from other traffic generators follow a hop by hop call setup procedure, trying to allocate bandwidth on each hop. By contrast, calls that follow the VP contest for bandwidth only at the VP source point. The capacity of the VP varies from 0 to 100 percent of the capacity of the smallest link on the VP (the MH->NWRK link which is rated at 45 Mbps). When the VP capacity is at 100% only calls from MHEX to RUTG are allowed, since all other calls find no available resources to proceed to their destination. When the VP capacity is reduced to 0, all calls are attempting the regular VC setup procedure.

Figure 12 shows the obtained measurements. The throughput curve reveals that maximum throughput is attained when the VP contract region is 30 percent of the link schedulable region. This happens because below that value, an increasing amount of call requests from MHEX to RUTG find the VP full and use the regular VC setup procedure, thereby forcing the signalling entity at MH and NWRK into an overload state, that causes high call setup times and higher blocking. When the VP

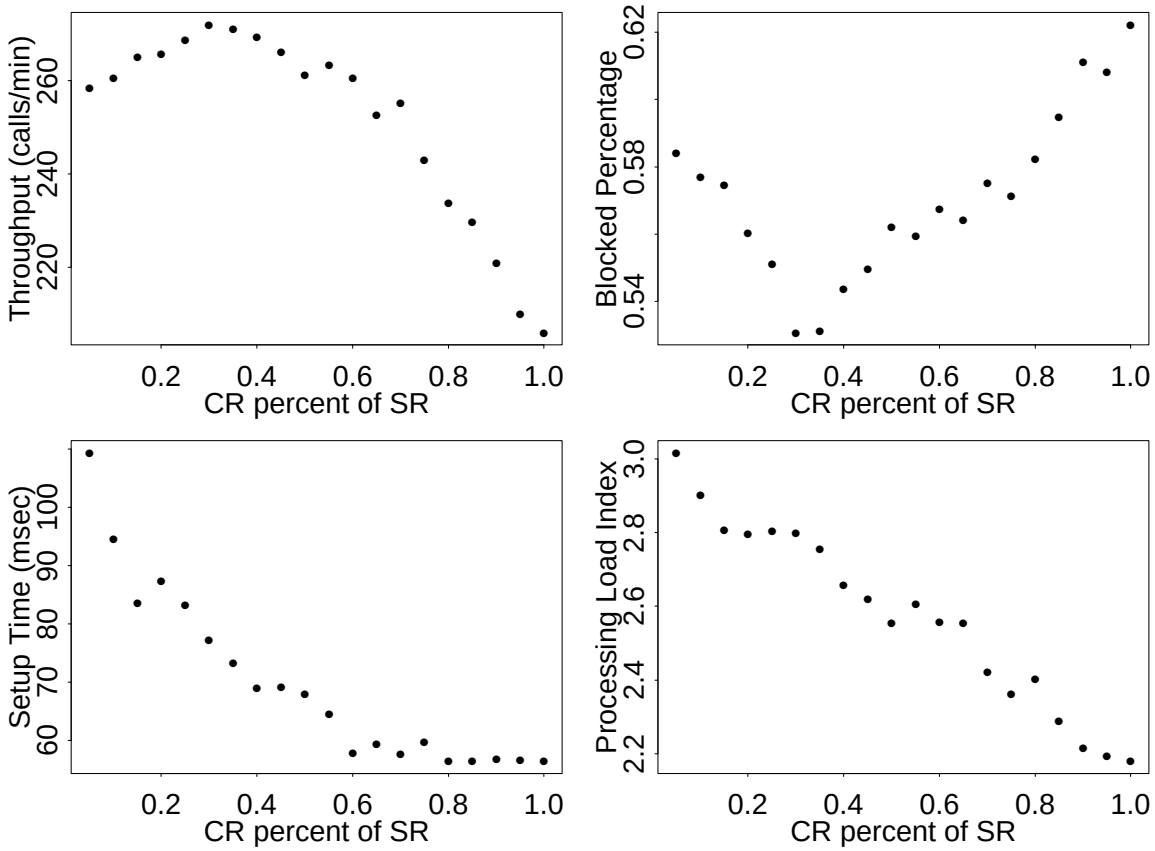

Figure 12: Virtual Path performance vs. allocated networking capacity

contract region increases above 30 percent, the throughput drops slowly as the extra capacity allocated for the VP is partially unused, and as a result a larger percentage of the interfering traffic (that does not follow the VP) is blocked. The fourth plot depicts the average number of signalling messages needed to establish (or reject) an incoming call. The numbers drop as the VP increases in capacity, as calls from MHEX to RUTG follow the VP and use less hops to reach the destination.

5. Summary and Future Work

A basic model for the performance management of VP connection services for ATM broadband networks was presented. A set of managed object classes following the OSI standard for network management with complete attribute structure was proposed. The call-level model enables the network manager to retrieve information from agents installed in the network, make decisions based on the configuration and performance observations, and apply a set of controls if the observed performance can be improved. These controls include setting the operating parameters of the signalling code and changing entirely or in part the distribution of the Virtual Paths in the system.

Our model was fully implemented on the Xunet ATM testbed. The manager is able to observe the call level performance of Xunet from a dedicated management tool. We have presented some aspects on the call level performance of Xunet and demonstrated the behavior of the network when Virtual

Paths are in use.

We are currently working on an algorithm for an automated tool that observes the offered call load and the call-level performance related properties and makes decisions regarding the VP distribution policy and the operating parameters of the signalling software. Such a system will significantly facilitate the performance management task for a network with a large number of nodes and Virtual Paths.

Acknowledgments

The authors would like to acknowledge the contribution of Giovanni Pacifici in configuring the experiments on Xunet. This work was funded in part by NSF Grant CDA-90-24735, and in part by a grant from the AT&T Foundation.

References

[ANE93] Nikos G. Aneroussis, Charles R. Kalmanek and Van E. Kelly, "Implementing OSI Management Facilities on the Xunet ATM Platform," in *Proceedings of the Fourth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Long Branch, New Jersey, October 1993.

[FRA92] A.G. Fraser, C.R. Kalmanek, A.E. Kaplan, W.T. Marshall and R.C. Restrick, "Xunet 2: A Nationwide Testbed in High-Speed Networking," in *Proceedings of the IEEE INFOCOM'92*, Florence, Italy, May 1992.

[HYM91] Jay M. Hyman, Aurel A. Lazar, and Giovanni Pacifici, "Real-time scheduling with quality of service constraints,"' *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1052-1063, September 1991.

[HYM93a] Jay M. Hyman, Aurel A. Lazar, and Giovanni Pacifici, "A separation principle between scheduling and admission control for broadband switching," *IEEE Journal on Selected Areas in Communications*, vol.11, pp. 605-616, May 1993.

[HYM93b] Jay M. Hyman, Aurel A. Lazar, and Giovanni Pacifici, "Modelling VC, VP and VN Bandwidth Assignment Strategies in Broadband Networks", *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video*, Lancaster, United Kingdom, November 3-5, 1993, pp. 99-110.

[ICM93] ICM Consortium, "Revised TMN Architecture, Functions and Case Studies", ICM Deliverable 5, 30 September 1993.

[IET94] Internet Engineering Task Force, "Definition of Managed Objects for ATM Management", Internet Draft Version 7.0, March 9, 1994.

[ISO88] Information Processing Systems - Open Systems Interconnection, "OSI Management Framework," October 1988. International Standard 7498-4.

[ISO91a] Information Processing Systems - Open Systems Interconnection, "Structure of Management Information - Part 4: Guidelines for Definition of Managed Objects," July 1991. International Standard 10165-4.

[ISO91b] Information Processing Systems - Open Systems Interconnection, “Systems Management - Fault Management - Part 5: Event Report Management Function,” July 1991. International Standard 10164-5.

[ISO92] Information Processing Systems - Open Systems Interconnection, “Systems Management - Performance Management - Part 11: Workload Monitoring Function”, April 1992. International Standard 10164-11.

[LAZ90] Lazar, A.A., Temple, A.D. and Gidron, R., “An Architecture for Integrated Networks that Guarantees Quality of Service”, *International Journal of Digital and Analog Cable Systems*, Vol. 3, pp. 229-238, April-June 1990.

[LAZ92] Lazar, A.A., “A Real-Time Management, Control and Information Transport Architecture for Broadband Networks”, in *Proceedings of the 1992 International Zurich Seminar on Digital Communications*, Zurich, Switzerland, March 1992.

[LAZ93] Lazar, A.A. and Stadler, R.,“On Reducing the Complexity of Management and Control in Future Broadband Networks”, in *Proceedings of the Fourth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Long Branch, New Jersey, October 1993.

[KNI91] George Pavlou, Graham Knight and Simon Walton, “Experience of Implementing OSI Management Facilities,” in *Integrated Network Management, II* (I. Krishnan and W. Zimmer, editors), pp. 259–270, North Holland, 1991.

[MAZ91] S. Mazumdar and A.A. Lazar, “Objective-Driven Monitoring,” in *Proceedings of the Second International Symposium on Integrated Network Management*, pp. 653–676, Washington, D.C., April 1991.

[NEU93] Neumair B., “Modelling Resources for Integrated Performance Management”, in *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management*, San Francisco, California, 18-23 April, 1993.

[OHT93] Ohta, S., and Fujii, N., “Applying OSI System Management Standards to Virtual Path Testing in ATM Networks”, in *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management*, San Francisco, California, 18-23 April, 1993.

[SAR93] H. Saran, S. Keshav, C.R. Kalmanek and S.P. Morgan, “A Scheduling Discipline and Admission Control Policy for Xunet 2", *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video*, Lancaster, United Kingdom, November 3-5, 1993.

[TSU92] Tsuchida, M., Lazar, A.A., Aneroussis, N.G., “Structural Representation of Management and Control Information in Broadband Networks”, in *Proceedings of the 1992 IEEE International Conference on Communications*, Chicago IL., June 1992.

Appendix A. A Generic Model for Cell Level and Call Level Performance Monitoring

The OSI reference model provides connection-oriented, connectionless and transaction-oriented modes of communication. In the ATM world, only connection-oriented communication is em-

ployed. [LAZ90] and [NEU93] define two different levels of refinement for performance management purposes: the call level and the cell level (Figure 13). On the first level, we consider the ATM

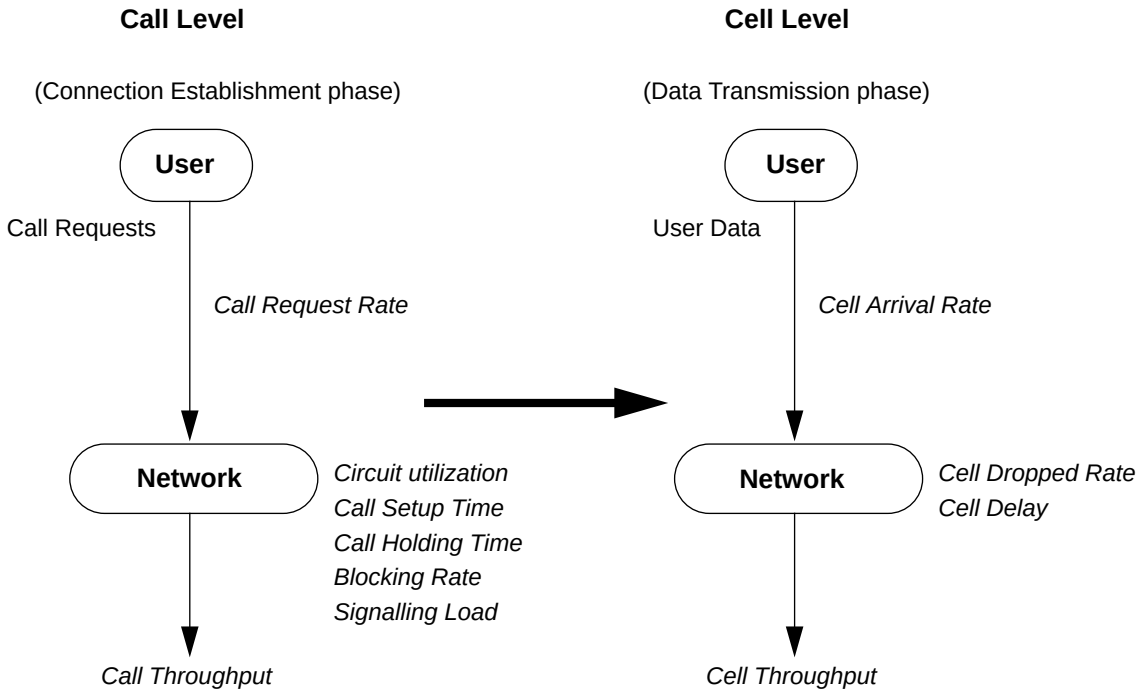

Figure 13: Performance Management Related Properties at the Call and the Cell Level

network as the service provider. The user makes service requests by attempting to establish ATM connections of a certain service class (e.g., voice, video, etc.) and of a certain networking capacity. This represents monitoring performance at the level of the C-plane. The performance related properties include the following:

- Call request rate: average number of connection requests in the unit of time
- Utilization and capacity: Number of currently or averaged open connections and maximum number of open connections allowed (due to limits in the available networking capacity or availability of circuits).
- Service Time: Average holding time of connections. The service time multiplied with the request rate gives the offered load in Erlangs. This quantity can be used for network capacity planning.
- Blocking (or Error) rate: The rate of calls being blocked due to resource unavailability or any other error. Further subdivision of this quantity into categories can give more information on the rates with which every error occurs. For example we can measure the rate with which connections are being blocked due to networking capacity unavailability, routing errors, circuit identifier unavailability, signalling errors, etc. The ratio of the error rate with the request rate gives the blocking probability.

- Signalling load: This quantity measures the load imposed on the signalling system, which is performing the connection establishment requests and depends on the request rate and the necessary time to complete the connection setup request. The higher the load, the less the ability of the signalling system to provide the service. Factors that contribute in increasing the signalling load are: high number of signalling message retransmissions due to signalling protocol errors, higher percentage of connection attempts over routes with many hops (that increases the number of transmitted messages), high call arrival rate, etc.

The second level of refinement for performance monitoring purposes refers to the data transmission phase of communication, after the connection has been established. The user and the network maintain the service user / service provider relationship but with different semantics: the user sends cells to the network for transmission. This represents monitoring performance at the level of the U-plane. The following performance related properties are of significance:

- Cell Arrival Rate: Average number of cells arriving from the user in the unit of time.
- Cell Drop Rate: Average number of cells that are being dropped in the unit of time.
- Cell Throughput: Average number of cells transmitted successfully.
- Cell transmission delay: Average delay for to transmitting of a cell from the time it has been received at the input port.

Appendix B. GDMO Definitions of Managed Object Classes

```
vcEntity MANAGED OBJECT CLASS
    DERIVED FROM top;
    CHARACTERIZED BY vcEntityPackage;
    CONDITIONAL PACKAGES cellPerformancePackage;
    REGISTERED AS { xunetManagedObjectClass 1 };

link MANAGED OBJECT CLASS
    DERIVED FROM top;
    CHARACTERIZED BY linkPackage;
    CONDITIONAL PACKAGES cellPerformancePackage, callPerformancePackage,
                         admissionControllerPackage;
    REGISTERED AS { xunetManagedObjectClass 2 };

sourceDestination MANAGED OBJECT CLASS
    DERIVED FROM top;
    CHARACTERIZED BY sourceDestinationPackage;
    CONDITIONAL PACKAGES cellPerformancePackage, callPerformancePackage;
    REGISTERED AS { xunetManagedObjectClass 3 };

virtualPath MANAGED OBJECT CLASS
    DERIVED FROM top;
    CHARACTERIZED BY virtualPathPackage;
    CONDITIONAL PACKAGES cellPerformancePackage, callPerformancePackage,
                         admissionControllerPackage;
    REGISTERED AS { xunetManagedObjectClass 4 };

connectionMgmt MANAGED OBJECT CLASS
    DERIVED FROM top;
```

```
    CHARACTERIZED BY connectionMgmtPackage;
    REGISTERED AS { xunetManagedObjectClass 1 };

################## PACKAGE DEFINITIONS ####################

vcEntityPackage PACKAGE
BEHAVIOUR vcEntityBehaviour;
ATTRIBUTES
    inputSlot GET,
    inputVPI GET,
    inputVCI GET,
    outputSlot GET,
    outputVPI GET,
    outputVCI GET,
    timeEstablished GET,
    trafficDescriptor GET,
    adaptationLayer GET,
    sourceService GET,
    destinationService GET;
REGISTERED AS { xunetPackage 1 };

linkPackage PACKAGE
BEHAVIOUR linkBehaviour;
ATTRIBUTES
    linkType GET,
    linkModuleDescription GET,
    linkSource GET,
    linkDestination GET,
    linkSlotNumber GET;
REGISTERED AS { xunetPackage 2 };

sourceDestinationPackage PACKAGE
BEHAVIOUR sourceDestinationBehaviour;
ATTRIBUTES
    sourceNodeAddress GET,
    destinationNodeAddress GET;
REGISTERED AS { xunetPackage 3 };

virtualPathPackage PACKAGE
BEHAVIOUR virtualPathBehaviour;
ATTRIBUTES
    vpIdentifier GET,
    vpSource GET,
    vpDestination GET,
    vpRoute GET.
    timeEstablished GET,
    circuitCapacity GET;
REGISTERED AS { xunetPackage 4 };

admissionControllerPackage PACKAGE
BEHAVIOUR admissionControllerBehaviour;
ATTRIBUTES
    networkingCapacity GET-REPLACE,
    admissionControllerOperatingPoint GET,
    admissionControlPolicy GET-REPLACE,
    admissionControllerOperationalState GET-REPLACE,
```

```
    admissionControllerAdministrativeState GET-REPLACE;
REGISTERED AS { xunetPackage 5 };

callPerformancePackage PACKAGE
BEHAVIOUR callPerformanceBehaviour;
ATTRIBUTES
    activeCircuits GET,
    callArrivalRate GET,
    callArrivedCounter GET,
    callResourceBlockedCounter GET,
    callErrorBlockedCounter GET,
    callBlockingRate GET,
    setupTime GET,
    holdingTime GET,
    numExchangedMessages GET,
    measureInterval GET;
REGISTERED AS { xunetPackage 6 };

cellPerformancePackage PACKAGE
BEHAVIOUR cellPerformanceBehaviour;
ATTRIBUTES
    cellTransmittedCounter GET,
    cellTransmissionRate GET,
    cellDroppedCounter GET,
    cellDroppedRate GET,
    avgCellDelay GET;
REGISTERED AS { xunetPackage 7 };

connectionMgmtPackage PACKAGE
BEHAVIOUR connectionMgmtBehaviour;
ATTRIBUTES
    signallingProcessingLoad GET,
    maxSignallingProcessingLoad GET-REPLACE,
    signallingRetransmitTimeout GET-REPLACE,
    signallingCallSetupTimeout GET-REPLACE;
REGISTERED AS { xunetPackage 8 };
```