INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

MANAGEMENT OF HIGH-SPEED OPTICAL ATM NETWORKS

by

GUOWEI ZHANG

Thesis submitted to
The Faculty of Graduate Studies and Research in
partial fulfillment of the requirements for
the degree of Master of Science

Department of
Systems and Computer Engineering
Carleton University
Ottawa, Ontario
Canada

© Guowei Zhang 1997

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-27063-7

The undersigned reommed to
The faculty of Graduate Studies
acceptance of the thesis

Management of High-Speed Optical ATM Networks

Submitted by **Guowei Zhang, B.Sc, M.Sc, and PhD**

in partial fulfillment of the requirements for
the degree of Master of Science

Chair

Systems and Computer Engineering

Department

Thesis Supervisor

Carleton University

September 18, 1997

ABSTRACT

The combination of ATM and optical networking technologies leads to a new generation of communication networks with flexible switching and high bandwidth capacity. This project proposes a TMN-like network management architecture to deal with the challenges of the ATM optical networks and the multi-vendor situation. The thesis discusses the functionality and interface modules of the network management system. To provide a platform independent GUI, a Web-based approach is suggested and discussed in detail. The target network is based on the SONET double rings and all-optical networks. The TDM and WDM technologies are used to shift the routing workload from ATM switches to the optical transport nodes and ADM is used as user access to the optical networks. A specific wavelength can be reserved for network management. In addition, some experimental studies have been conducted on the software development to support the proposed management architecture. The thesis describes the DPI/DMI agent architecture and how to implement a component, and introduces the CMU/UCD SNMP package and experiences in using it. It also presents two Client/server systems written in sockets and ATM-APIs to evaluate the management communications in the network management system.

ACKNOWLEDGMENTS

I would like to extend my sincere gratitude to Professor B. Pagurek, my supervisor, for providing me the unique study opportunity, for his helpful guidance and encouragement, and for his financial support in the form of a Research Assistantship during the course of the study.

As for the collaboration research in NRC, I wish to thank Dr. J. Zhou for leading me to the exciting photonics field and for countless discussions. I am grateful to Dr. Suhayya Abu-Hakima and Dr. Jacek Chrostowski for providing me the research opportunity in their groups.

May I also acknowledge the financial support of Carleton University in the form of scholarship.

Many thanks to Cheryl Schramm for correcting my English in the thesis.

I wish to thank all my friends and relatives for their long-time support and encouragement.

Finally, I would like to thank my wife Jingling for her endless support, patience, and enduring love.

Dedicated

to my Mom

to whom I am eternally grateful

and

To my Dad

who is always there in my heart

LIST OF ACRONYMS

AAL	ATM Adaptation Layer
ACK	(Positive) Acknowledgment
ACSE	Association Control Service Element
ADM	Add/Drop Multiplexer
AON	All-Optical Network
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
ATMOS	ATM Optical Switching
B-ISDN	Broadband Integrated Services Digital Network
BML	Business Management Layer
BSD	Berkeley Software Distribution
CERN	European Center for High Energy Physics
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element
CMU	Carnegie Mellon University
CORBA	Common Object Request Broker Architecture
DCE	Distributed Computing Environment
DCN	Data Communication Network
DEC	Digital Equipment Corporation
DLL	Dynamic Linked Libraries
DMI	Desktop Management Interface
DMTF	Desktop Management Task Force
DPI	Distributed Protocol Interface
EDFA	Erbium doped fiber amplifier
EL	Element Layer
EML	Element Management Layer
Gbps	Gigabit per second

GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
IDL	Interface Definition Language
IP	Internet Protocol
ITU	International Telecommunication Union
ITU-U	ITU Telecommunication Standardization Sector (formerly CCITT)
LAN	Local Area Network
MAC	Media Access Control
MAN	Metropolitan Area Network
Mbps	MegaBit Per Second
MD	Mediation
MIB	Management Information Base
MIF	Management Information Format
MIT	Massachusetts Institute of Technology
MTU	Maximum Transfer Unit
NCSA	National Center for Supercomputing Applications
NE	Network Element
NM	Network Management
NML	Network Management Layer
NMS	Network Management System
N-PDU	Network Protocol Data Unit
NRC	National Research Council Canada
OAM	Operations and maintenance
OID	Object Identifier
OS	Operation System
OSF	Open Software Foundation
OSI	Open Systems Interconnection
OT	Optical Terminal
PCM	Pulse Code Modulation
PDU	Protocol Data Unit

PVC	Permanent Virtual Circuit
QA	Q-Adapter
QOS	Quality Of Service
RACE	Research and Development in Advanced Communications in Europe
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SDH	Synchronous Digital Hierarchy
SML	Service Management Layer
SNMP	Simple Network Management Protocol
SONET	Synchronous Optical NETwork
SSCOP	Service Specific Connection Oriented Protocol
SPE	Synchronous Payload Envelope
STM	Synchronous Transfer Mode
STS	Synchronous Transport Signal
SVC	Switched Virtual Circuit
TCP	Transmission Control Protocol
Tcl/Tk	Tool Command Language/Toolkit
TDM	Time Division Multiplexing
TMN	Telecommunications Management Network
UDP	User datagram Protocol
UCD	University of California at Davis
UNP	Underlying Network Protocol
VC	Virtual Channel
VCI	Virtual Channel Identifier
VP	Virtual Path
VPI	Virtual Path Identifier
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing
WS	Work Station
WWW	World Wide Web

CONTENTS

1. INTRODUCTION	1
1.1 Optical ATM Networks	3
1.2 Next-Generation NMS	6
1.3 Scope of the Thesis	9
2. MANAGEMENT PROTOCOLS AND TOOLS	12
2.1 Management Protocols and TMN	12
2.1.1 SNMP	12
2.1.2 TMN and CMIP	14
2.2 DMI SNMP Agent	19
2.2.1 DPI/DMI agent architecture	21
2.2.2 DMI API	24
2.2.3 DMI MIF	26
2.2.4 Optical switch agent	28
2.3 SNMP Manager	30
2.3.1 CMU/UCD package	30
2.3.2 MIB Tree and query example	31
3. MANAGEMENT COMMUNICATIONS	34
3.1 Network APIs	35
3.1.1 Client/server model	35
3.1.2 BSD Socket	36
3.1.3 ATM API	38
3.2 Client/Server Performance	39
3.2.1 Experimental setup	40

3.2.2 Case 1: Normal behavior	41
3.2.3 Case 2: Abnormal behavior	43
4. OPTICAL ATM NETWORKS	50
4.1 ATM Network	51
4.2 Optical Transport Network	55
4.2.1 SONET	56
4.2.2 Optical networking	59
4.3 Optical ATM Network Architecture	62
5. MANAGEMENT ARCHITECTURE	67
5.1 Basics	67
5.2 Management Framework	69
5.3 Interface Modules	73
5.4 Network Managers	76
5.4.1 Connection manager	76
5.4.2 Fault manager	81
5.4.3 Performance manager	82
5.5 WWW and GUI	83
5.5.1 Fundamentals	83
5.5.2 Software components	85
6. CONCLUSIONS AND FUTURE WORK	88
6.1 Experimental Results	88
6.2 High-Speed ATM Optical Networks	91
6.3 Next-Generation NMS	92
APPENDICES	95
A.1 Optical Switch MIF	95
A.2 SystemView Agent Kit Set Up	104
A.3 HTTP	105
REFERENCES	108

Chapter 1

INTRODUCTION

In communication networks, existing and new services keep growing and will continue to do so in the coming years. Examples of services are high-speed file and data transfer and multimedia applications in the form comprising of data, voice, and video. These services are very diverse in terms of bandwidth capacity, channel occupancy, and the means of connection set-up (packet switching or circuit switching). It seems that the transport of such high-speed network traffic will rely on Asynchronous Transfer Mode (ATM) standard due to its flexible multiplexing and routing advantages, and its wide acceptance.

Unfortunately, routing and multiplexing in ATM networks are performed electronically and this ultimately is a limitation to the deployment of a high-speed network. Optics is used as the transmission media since the bandwidth offered by optical fibers is virtually unlimited. Yet because of the bottleneck of the electronic switching, currently available ATM switches can only provide bandwidth capacities at a few Gbps in a single fiber, far below the fiber's potential capacity. Thus future broadband transport networks will need to avoid the shortcomings used

by the electronic switching implementations. To this end, photonic technologies will play a more important role in the network implementations than ever before.^[1] Wavelength division multiplexing (WDM), for example, has the ability to increase the bandwidth capacity beyond 10 Gbps. As the number of network service subscribers grows and traffic volume increases, the implementation of an efficient access and tandem network requires switches with aggregate capacities in hundreds of Gbps.^[2] Therefore a hybrid network architecture composed of ATM and optical technologies is likely a practical solution. In this approach, optical networks transport multiplex high-data-rate traffic in WDM or time division multiplexing (TDM), and ATM switches are used as the relatively low-speed and access nodes.

In a traditional ATM network, a multihop connection traverses a number of stages of optical-to-electronic conversion, electronic demultiplexing, and low speed cross connection. In an optically routed multiwavelength network, traffic is aggregated into a high-capacity fiber in order to reduce the employment of the electronic equipment significantly. However, the management of these high-speed optical ATM networks is a very challenging topic.

It should be pointed out that several technologies such as frame relay may be used to transport high-data-rate traffic. However, ATM is of special interest for this thesis for two reasons. Firstly, ATM is widely deployed and has wide acceptance. Secondly, the focus of the thesis is high-level network management (NM), not the transport layer protocol. Thus ATM is chosen as the target high-data-rate protocol during the discussions in the thesis.

1.1 Optical ATM networks

Advancements in high-speed optical networking technologies follow two directions. In the first, ATM remains the base technology and the research effort is to develop optical or semi-optical ATM switches to overcome the electronic switching bottleneck. This approach recognizes that ATM is a widely accepted networking standard for which there is widespread development activity. In the other direction, many projects are being conducted in order to explore technologies of all-optical networks (AONs).

In an effort to advance ATM optical switching technologies, the consortium Research and Development in Advanced Communications in Europe (RACE) launched a project, namely ATM optical switching (ATMOS), to investigate the space and wavelength domains for fast optical routing and buffering in ATM networks. In their corporate optical ATM network architecture (Fig. 1.1), ATM electronic switches are interconnected with an optical double ring through optical access nodes.^[1,3] The access nodes perform such functions as to transfer ATM streams between the fiber bus and a specific output such as ATM cell hub or ATM switch, to synchronize the optical signals in the rings, and to demultiplex the traffic stream coming from the high-speed optical channels. ATM cells are inserted and dropped bit by bit using multifunctional high-speed switches with dedicated drivers in the cell processor block. Since only the space domain is exploited, ATM cells are electrically buffered at the lower-speed side before being inserted into the rings. This is in fact an add/drop multiplexer (ADM) architecture.

Figure 1.1 Optical access node.

Regarding the studies of optical transport networks, a consortium of wideband AONs was established by AT&T, DEC, and MIT in 1993.^[4,5] Its aim is to develop WDM and TDM optical networks with a capacity of 100 Gbps. The testbed is a 20-channel WDM system with a data-rate-per-wavelength ranging from 10 Mbps to 10 Gbps. As shown in Fig. 1.2, the architecture is a three-level hierarchy of subnetworks, which are denoted as L0, L1, and L2. Each of the subnets is an AON and capable of autonomous operations. At the bottom, a L0 is a high-performance LAN with several optical terminals (OT's) attached to it. Several L0s combine to form a L1 subnet which is in fact a MAN. At the top, a L2 is a WAN which is used to interconnect the L1s. Different wavelengths are employed at each level. A L0 broadcasts all received wavelengths coming from L1 and L2. Each OT listens for its own wavelength and performs the multiplexing and demultiplexing for the users.

Figure 1.2 All-optical network architecture.

This testbed uses an in-fiber, out-of-band signaling network to perform the network control, management, and synchronization. The control system has a node at each OT and at each network node. The signaling data travels in the fiber^[6] within a bandwidth of 10 Mbps between an OT and an L0, and 1.5 Mbps between nodes of L1 and L2. The signaling channel is an ATM and synchronous optical network (SONET) transition. A NM agent is implemented at each node of the network.^[6] The agent manages the high speed optical components of the node or terminal via a proxy using the standard NM protocol Simple Network Management Protocol (SNMP).

1.2 Next-Generation NMS

Optical ATM networks bring more challenges to the network management system (NMS) than do traditional networks. In the first instance, ATM uses one technology for transmitting diverse traffic types (data, voice, and video) and running different applications across LANs and WANs. Network administrators need an end-to-end overview of a variety of conditions, from throughput to cell delay, for tens of thousands of virtual connections. They must be able to monitor and control the physical-switching infrastructure. The use of ATM over optical transport networks adds another dimension to the NMS. Distribution of the wavelengths in WDM and channel assignment in TDM should be reconfigurable in order to serve different purposes. For example, the Olympic games take place in a city for a specific period of time, during which network traffic increases by hundreds and thousands of times the usual traffic. During this time period, the wavelengths in the WDM system are thus reconfigured to assign more bandwidth to the city.

None of the major SNMP-based platforms offer the bulk protocol capabilities needed to manage the large-scale optical ATM networks. Moreover, in a multi-vendor environment the NMS must integrate SNMP, Common Management Information Protocol (CMIP), and proprietary protocols since it is impossible to require all network devices to support the same NM protocol.

Project ATDNet was conducted by Bells (Bell Atlantic Federal Systems, Bell Atlantic, and Bellcore). Among various attempts, it is a good example in building the next generation NMS for optical ATM networks.^[7,8] Their efforts mainly include NM, ATM signaling, and traffic management of ATM/SONET corporate networks. The network backbone is connected

by OC-48 ADMs to form a bi-directional SONET ring, giving a capacity of 2.5 Gbps. The Telecommunications Management Networks (TMN) architecture was adopted as a functional framework of the NMS. ATDNet is a good starting point in the design of a NMS for optical ATM networks.

The following paragraphs describe the main characteristics of the next-generation NMS. The issues discussed here include management integration capabilities, multitiered and multiprotocol architecture, distributed computing, object-oriented design, applying Common Object Request Broker Architecture (CORBA) to NMS, and Web-based NMS. ATDNet NMS will be compared wherever it is applicable.

An integrated NMS should provide the capabilities to manage the network across the ATM and the optical layers. End-to-end connections are made in one step. The system automatically routes the connections over an optical path. In ATDNet, a graphical user interface (GUI) is provided to display an integrated view of network maps, equipment faceplates, and alarm status. Integrated fault management is achieved by correlating alarms from SONET and ATM layers. Faults are displayed by coloring the particular SONET link, the fiber connecting the ATM switch and the ADM, and the ATM or SONET circuit pack that has failed. However, it is obvious that this NMS targets only the ATM/SONET networks. In optical networks, many other techniques such as WDM may be used and thus the NMS must provide the capability to manage them.

Multitier and multiprotocol management architecture supports management scalability and interoperability. Networks are partitioned into subnetworks, with each managed by a separate system. Subnetwork management systems are designed to encapsulate manufacturer-

specific information models and protocols. Network-level management processes provide end-to-end NM capabilities across multiple subnetworks. Therefore a TMN-like functional hierarchy should be employed in order to split the functionality between modules “vertically”.

Distributed computing is a specific characteristic to the NMS for large-scale networks. Management modules may run on different computers in different platforms and use different protocols. The quality of the management communications in terms of reliability, robustness, and security has a significant impact on the performance of a NMS. In ATDNet, Distributed Computing Environment (DCE) and Remote Procedure Call (RPC) were used for management communications. Different module interfaces were implemented to translate the management data between the protocol-specific formats and the internal data mode. For large and complex optical ATM networks, a NMS should adopt a standard communication platform architecture such as CORBA, in order to deal with the multi-vendor and heterogeneous computing environment. In the meantime, client/server systems written in different network application programming interfaces (APIs) and agent/subagent systems should be investigated to ensure the quality of the management communications and proper distribution of the NM functionality.

Advanced software design is a key step in developing a powerful NMS for large systems such as optical ATM networks. NMSs should employ object-oriented technology to achieve software reusability and modularity, as is the case in ATDNet. Management applications, such as configuration management, fault management, and performance management, were developed by using building blocks and well-defined interfaces to separate the specification

from implementation. Thus a TMN-like functional hierarchy again is required to distribute the management functionality in the “horizontal dimension”.

CORBA provides a standard environment for distributed objects management. The definition of the interfaces in interface definition language (IDL) separates the interfaces from their implementations. For a NMS, this means that management functions can be written in different languages, such as C++ and Java, on different platforms, and with various management protocols (SNMP, CMIP, and proprietary protocols). CORBA provides distributed computing capability to transport management messages transparently and to integrate diverse network entities in a heterogeneous environment. CORBA is a necessary computing environment in building a NMS for complex optical ATM networks.

The GUI is another key component in the NMS. Tool Command Language and Toolkit (Tcl/Tk) was used to implement the GUI in the ATDNet NMS. Tcl/Tk is like glue which can easily accomplish the software integration. However, it is not platform-independent. If a GUI is written for one platform, it might not work for other platforms. Web technology is a good candidate for a GUI implementation. A Web-based NMS provides a platform-independence. A network administrator can access the NMS from anywhere on any platform. When Java applets are involved, various management applications can be easily developed.

1.3 Scope of the thesis

The research topic of this project is the management of high-speed optical ATM networks. Because it is not realistic to implement a NMS or even to give a specification, the project focuses on the NMS architecture, i.e., a functional hierarchy framework. It identifies

practical issues in building a NMS. The thesis presents the research results in four parts: a study of evaluation of management tools, an analysis of the performance of the management communications, an exploration of the high-speed optical ATM networking technologies, and a proposal for the NMS architecture. The thesis is organized as follows.

Chapter 2 presents the experimental studies on some NM tools in support of the proposed management architecture. It briefly describes the NM protocols SNMP and CMIP and the TMN standard and illustrates the Distributed Protocol Interface (DPI) and Desktop Management Interface (DMI) agent architecture along with some experiences in building a subagent. The CMU/UCD SNMP manager package is introduced and the SNMP MIB subtree involved in the project is presented with some query examples.

Chapter 3 explores some agent and manager software development technologies. The chapter discusses the network APIs, client/server systems, and their performance. It introduces the basic client/server models and the network protocol hierarchy in the Internet domain. BSD sockets and the FORE System's ATM-API are compared. It presents the results of a performance analysis of client/server systems. The results provide guidelines for the design and implementation of a NMS for the high-speed optical ATM networks.

Chapter 4 gives a general overview of the issues in design of high-speed optical ATM networks and tries to provide a solid technical background for the NMS. It introduces ATM technology and the advantages of the optical networks. Finally an optical ATM network architecture is discussed.

Chapter 5 describes the fundamentals of the proposed NMS. A TMN-like functional hierarchy is suggested. The topics discussed include the integration of management protocols,

management communications, management object distribution and functionality modulation, introduction of CORBA to the NMS, network device autodiscovery, and Web-based GUIs.

Chapter 6 presents the conclusions and the future work. Technical details about software usage and setup are available in the appendices.

Chapter 2

MANAGEMENT PROTOCOLS AND TOOLS

This chapter begins with an overview of the two widely accepted NM protocols SNMP and CMIP and the standard TMN. §2.2 describes the DPI/DMI agent architecture and some experiences in building a subagent for an optical switch. §2.3 introduces the CMU/UCD SNMP manager package. The last section presents the SNMP MIB tree and some query examples.

2.1 Management Protocols and TMN

2.1.1 SNMP

SNMP is an application-level protocol in the TCP/IP protocol suite. Fig. 2.1 shows a typical configuration of the SNMP protocol^[9]

A minimal SNMP system consists of two parts: the manager and the agent. A manager process running on the management station maintains the database of the network inventory,

holds a hash table for mapping of object identifier (OID) and the names of the variables in the Management Information Base (MIB), and provides an interface to the network operator. The agent process is a daemon which controls the agent's MIB. It keeps running all the time to listen for requests from managers and interprets the SNMP messages. The agent device must also implement UDP and IP. The manager and the agent send and receive NM messages by using SNMP, which is implemented on top of UDP/IP and the underlying network protocol (UNP) such as Ethernet.

Figure 2.1 Configuration of SNMP.

Usually, the manager and the agent communicate in a client/server model. The manager sends a *get-request* to the agent for information and the agent replies with a *get-response*. The manager can also send *set-request* instructing the agent to update some piece of information with the value provided. An agent can also be configured to provide some specific information to the manager in an autonomous way by sending a *trap*.

In this thesis, a process that performs all basic SNMP functions and communicates directly with the SNMP agent is called a protocol manager, in order to distinguish the manager from other management processes. An agent that directly responds to the protocol manager is called a protocol agent. The terms protocol manager and manager can be interchanged only when no confusion is possible. This is also true for the protocol agent and agent. For example, at the application level, a process that performs diagnosis of alarms is called a fault manager. When diagnosing faults, this manager needs to call the SNMP protocol manager to retrieve relevant data. It would be confusing if one simply talked about a “manager”.

2.1.2 TMN and CMIP

Since the early 1980s, the standards bodies have been specifying a set of TMN guiding principles. These principles aim at being applicable across telecommunications technologies. They recommend the use of logically separate management networks (managing networks) to manage telecommunications networks and elements (managed objects), communicating via well defined, standardized interfaces.^[10] The architecture of TMN identifies the different

types of nodes and the interfaces between them.^[11] So far five TMN nodes and four interfaces have been identified, as illustrated in Fig.2.2 and described below.

Figure 2.2 A simplified TMN physical architecture.

The TMN nodes include the operation system (OS), Mediation (MD), Q-adapter (QA), workstation (WS), and network element (NE).

The OS represents the supervisory or control systems. OS's can be interconnected to form management hierarchies. OS functionality can be layered into four levels: element

management layer, network management layer, service management layer, and business management layer, as discussed below.

The MD may provide storage, adaptation in horizontal dimension, or filtering operations on data. However, this node is not well defined. In fact, MD is often referred as Q-adaptor (QA) in telecommunication industry.

The QA is used to connect a TMN management system to a non-TMN system, i.e. the managed telecommunication network. In reality, it is very difficult to develop QA's due to the problems in mapping between the TMN interfaces and the pre-existing interfaces.

The WS really stands for the TMN-user interface. It provides the presentation function to the user. It has a different meaning than the workstation of the computer world.

NE is the only node actually residing in the managed network and its primary function is to handle the user traffic, but not management information. The NE is the ultimate target of the management supervision and control.

The nodes identified above communicate with each other through the Data Communication Network (DCN). DCN is the transport network for TMN management messages. It was assumed to be an independent network from the telecommunication network. However, this restriction is relaxed in consideration of the costs.

The four TMN interfaces include Q3, Qx, F, and X. The Q3 connects an OS and any other node (NE, QA, MD, or OS in the same TMN). The Qx is like Q3, but with less functionality. The X interface is used for communications between OS's belonging to different TMN's, or between a TMN OS and a non-TMN OS that supports a TMN-like interface. The F interface is used for communications between a WS and any other nodes.

TMN's functionality can be viewed in a hierarchy structure. Five TMN layers have been identified in M.3010:^[11] the element layer (EL), the element management layer (EML), the network management layer (NML), the service management layer (SML), and the business management layer (BML). For resource management, only the first three layers are relevant, and thus the SML and BML are not discussed in this thesis.

The EL performs basic management functions for network devices. A network element is the physical device and the management process (e.g., SNMP agent process) running on behalf of the device.

In EML, network elements of a similar type are grouped together and managed as a subnetwork. Normally devices from the same manufacturer are aggregated into subnetworks. The EML may filter message traffic going to the NML so as to reduce the workload in NML.

The NML provides a management view of the whole network that is under one administrative domain. Through the EML, the NML can manage subnetworks or elements based on the view presented by the EML. NML can provide a detailed view of portions of the whole network. From NMS project's perspective, interesting functions include configuration management, fault management, and performance management. The TMN architecture allows the NMS to achieve a separation of concerns and to add a greater degree of modularity on NM applications. For example, the EML can use the same code to manage devices of the same type and the NML can use a protocol-independent model to facilitate a multi-vendor environment.

The decision by the standards bodies to allocate functionality to the EL, EML, and NML is consistent with trends in products currently available. Network equipment often comes with a vendor-specific management system. Sometimes a single system may contain functionality

of more than one layer. For example, every FORE System's ATM switch comes with a SNMP agent and probably with a management workstation.

Besides, the decomposition of the TMN functionality also reflects the production process. In a software development process, many people write software components which are related to the NM. They are divided into three groups: an equipment software group that develops the protocol agents for the equipment, a data providing group that writes software to retrieve data from the agents and send the data to the database, and the system management group that implements application algorithms and GUIs. The TMN layer structure is consistent with the work assignment scheme.

Adaptation of a protocol-specific information to a TMN system is a very critical step. Usually the TMN interface Q3 is implemented between NE and EML, and between EML and NML. At NML, a protocol-independent MIB is specified to ensure that information from different protocols such as CMIP and SNMP are compatible to facilitate interworking. The protocol-independent representation of the MIB address functions of the network management. A certain kind of agent (CMIP or SNMP) is implemented on every NE.

There are two major candidates for the management protocol: CMIP and SNMP. On one hand, to meet the requirement that the TMN interfaces be generic and flexible, the OSI system management technology was selected as the basis for the TMN interfaces. In OSI system management, the manager and the agent communicate using a seven-layer protocol suite. A key element of the suite for NM is the Common Management Information Service Element (CMISE) seated at the application layer. CMISE consists of a service definition, the Common Management Information Service (CMIS), and the CMIP. By adopting CMIP, TMN has gained reliable and robust communications capabilities, and a wealth of application-layer

building blocks which include the Association Control Service Element (ACSE) and the CMISE.

On the other hand, although SNMP is not considered a candidate in TMN standardization, its position in the marketplace ensures that it will be used in networks that are deployed. SNMP capabilities may be entirely adequate and more cost-effective for smaller devices, while the capabilities of CMIP will be very attractive for larger systems. It is believed that both CMIP and SNMP will exist in large networks.

Some conclusions can be made from above presentations of the SNMP and TMN. The SNMP is a widely accepted NM protocol for data communications that is relatively simple and does not need a very high speed NMS. Most devices in data communications networks support SNMP. Enterprise networks deal mainly with the data communications even though ATM is meant to integrate the data, voice, and video into a single network. Therefore it makes more economic sense to use SNMP for data communications NMS. However, this does not spell the end of TMN. In fact, it may be desirable to extend TMN to include SNMP or to use the functional framework of TMN to build up private NMS's. This approach is especially helpful for a heterogeneous environment.

2.2 DMI SNMP Agent

As mentioned above, a network device usually comes delivered with their own protocol agent, interfaces, and sometimes a whole management system. However, for non-standard, small devices, and experimental devices, the user very often has to write his/her own software to incorporate a device into the NMS. Sometimes, customers want a standard agent installed on

the equipment, instead of a proprietary agent. An agent development tool is therefore essential for building a NMS.

During the course of the project, the implementation of an SNMP agent on a NT workstation was demanding and critical. The agent monitors and controls a non-standard optical crossconnect switch that is a core device in the optical network. It requires high extensibility and ease of implementation and instrumentation.

Within the Photonics laboratory, different devices such as optical crossconnect switches and amplifiers are to be prototyped and tested on the same physical network setup. Due to its dynamic research characteristics, this laboratory accommodates many co-researchers every year; for example, Co-op and summer students, graduate students, and visiting scholars. Not all of them are expected to be familiar with NM protocols. An extensible agent framework is therefore in need for various NM purposes.

The IBM's SystemView DPI/DMI Agent kit is an agent framework and is well suited for this situation. It was installed successfully on machine *rho* in the laboratory. DMI was proposed by Desktop Management Task Force (DMTF).

In addition to fulfilling the minimal requirements mentioned above, the DPI/DMI architecture has some advantages over a conventional agent implementation. It allows managed resources to be instrumented once, but managed by any DMI-enabled manager, even those DMI-enabled managers that use different management protocols (such as SNMP, CMIP, CORBA, and private management protocols), although the package is currently implemented only for SNMP. With the DPI agent/subagent structure, many subagents can be attached to the master agent locally or remotely. This is convenient for a working environment where workstations are for multiple users and multiple purposes. Each user can

implement a separate subagent for a specific purpose. Remote subagents can help to distribute the NMS functionality.

This section introduces the DPI/DMI agent kit and presents some experiences in building SNMP subagents. The information about the setups of the package is presented in Appendix A.2.

2.2.1 DPI/DMI agent architecture

Fig. 2.3 shows the architecture of the DPI/DMI SNMP agent package. This system is a set of functions and agents that provide flexible management of the applications and resources. By acting as an SNMP agent, it provides access to system components that have been defined according to the DMI standard and access to the DPI subagents.

The DPI-enabled SNMP master agent acts on behalf of all subagents which are registered to it. It handles SNMP requests and responses between management applications and the subagents through DPI. The *SNMPD* executable file installed with the package replaces any *SNMPD* file (such as Microsoft extensible agent) that might already be installed in the computer.

The DPI SNMP master agent is an extension of the SNMP agent that enables the user to dynamically add, delete or replace management variables in the local MIB without recompiling the agent system.^[12,13] A subagent can register a MIB variable or subtree with the master agent. Different subagents are distinguished by being assigned different OID. Any request received by the master agent for a registered variable is passed through the DPI to the

Figure 2.3 Architecture of the proxy agent for the optical switch.

subagent which serves the request and returns a response to the master agent. In principle, the management stations do not need to know whether the master agent calls on other processes to obtain an answer. As far as they can tell, there is only one NM agent, i.e., the master agent, running on the host.

The DPI agent/subagent can reduce the processing overhead significantly. For instance, consider that there are several devices in a remote location to be managed. In a conventional approach, each of the devices would have its own separate agent. If the manager requests the values of a certain variable for all these devices, it would have to make one request for each device. This would increase the traffic burden for the network and the time delay for the request significantly. In the DPI agent/subagent structure, the manager makes only one request to the master agent, and the master agent collects all necessary information from the subagents, packs the data into a single packet, and sends the aggregate information back to the manager. For example, one can use *snmpwalk*, as described in §2.3.1, to query the data. This can reduce the request overhead and the time delay. The trick is that SNMP can “walk” across different OIDs within one host (IP) as is the case in DPI master/subagent, but can not do it across different IPs as would be the case if the DPI subagents were independent agents.

The package provides two subagent interfaces: DPI and DMI. A DPI subagent is a standalone subagent and can be installed locally or remotely with respect to the master agent. This may be used to build a distributed NMS.^[14] During the course of the project, the SNMP DPI subagents have been successfully installed and tested.

A DPI/DMI subagent handles DPI requests from and responds to the master agent. When it receives a request, the subagent translates the MIB variable into a corresponding

DMI MIF (Management Information Format) attribute defined in a DMI component. The result of the request is sent to the management application in the reverse procedure. In addition, indications sent by DMI components in the system are converted to SNMP traps and sent to the management applications by the master agent. MIF attributes are active only when they are registered by the DPI/DMI subagent with the master agent. In principle, the DMI subagent could be installed locally or remotely with respect to the master agent. In short, a DPI/DMI subagent serves as a translator between the DPI-enabled SNMP master agent and the DMI components.

It is interesting to compare the two subagent APIs. A DPI subagent has a lighter processing overhead than the DMI equivalent does; however it must be implemented in a protocol-dependent manner. If a different protocol is to be supported, the subagent has to be rewritten completely. In the DPI/DMI approach, a subagent is actually a gateway or interface that is well separated from the implementation. To support a new protocol, all that is needed is to rewrite the interface to the upper layer; the implementation of the components (i.e., DMI instrumentation) remains the same. This provides the capability for the code reuse and can significantly reduce the software development cost.

In the course of the project, DMI API was used to implement a subagent for the optical crossconnect switch. For a better understanding, more explanations about DMI are given below.

2.2.2 DMI API

In the context of DMI, a component is a physical or logical element of a system. An element could be a piece of hardware or software. The program code that directly manages

the component is referred to as the component instrumentation. Information about a component is defined in the specific MIF file which is similar to, but not exactly the same as, the MIB in the context of SNMP. Some details of MIF are presented in §2.2.3 and Appendix A.1.

The DPI/DMI agent kit is available for three platforms: OS/2, AIX, and Windows NT/95. It was successfully installed on the Windows NT3.51 workstation *rho* in the Photonics laboratory. The NT machine came with an extensible Microsoft SNMP Agent that is the interface to the Microsoft DLLs (Dynamic Linked Libraries) for MIB-2 support. The Microsoft agent as an independent agent was disabled before the DPI master agent was installed and then was attached as a subagent to the master agent.

As an application interface, the DMI MIF browser can access the functions that the DMI service layer provides. The browser is used to view the values of the instances of each component that has been installed, and to install/uninstall components. By using the browser, one does not need a SNMP manager to operate the components.

To enable the mapping of a component by the DMI subagent, the component's MIF file must be registered with the SNMP OID tree. The mapping information is kept in the file *DMISA.MAP* in which each component has an entry. For example, the MIF file *RS232DIR.MIF* developed for the optical switch was assigned an entry as follows

```
"1.3.6.1.4.1.412.99.1.5" 1 1 1 1 "Optical Switch - Direct  
Interface" 0 0
```

where 412 is the enterprises OID of DMTF and 99 was arbitrarily assigned for the cooperative project between Carleton University and NRC. The MIF-to-MIB utility

(*MIFTOMIB*) provided with the package was used to generate an SNMP MIB from the MIF file. The following converts the MIF file *RS232DIR.MIF* to the MIB file *OPSWITCH.MIB*

```
D:\SVA\DMI\BIN\OS2\MIFTOMIB \
  "switchGroups = {enterprises dmtf(412) Carleton-NRC(99)
dmiMib(1) 1}" \ D:\SVA\DMI\EXAMPLES\RS232DIR\RS232DIR.MIF \
  D:\USERS\ZHANG\nOTES\OPSWITCH.MIB
```

so that the MIB's path to the SNMP OID tree is specified as:

```
enterprises.dmtf.Carleton-NRC.dmiMib.switchGroups.
```

The corresponding OID is

```
.1.3.6.1.4.1.412.99.1.1
```

2.2.3 DMI MIF

The MIF file is an ASCII file that defines the elements of a component to the DMI. The DMI service layer maintains all installed MIFs in a MIF database. Unlike the SNMP MIB, where ASN.1 is used, MIF uses its own format or syntax. This subsection briefly describes the relevant MIF syntax.

The MIF structure is composed of several main definitions: *Component*, *Path*, *Group*, and *Attribute*. All definitions are in the form of

```
Start name
    definition contents here
End name
```

where *name* is one of the definitions.

A *Component* is the highest level definition in the MIF file. All other elements of the MIF file are defined within the scope of a component. There can be only one component definition in a MIF file.

A *Path* specifies the pathname of the instrumentation code. There are two kinds of programs: *Overlay* and *Direct-interface*. An *Overlay* program is invoked by the service layer whenever a management application requests to retrieve or set the value of an attribute in the component. For a Windows NT/95 system, *Overlay* programs are written as DLLs. A *Direct-interface* program runs continuously as a separate process in the system. When the service layer receives requests from management applications, it activates and passes the request to the *Direct-interface* program, which returns the appropriate values.

A *Group* arranges one or more attributes into logical sets. It can be used to represent tables, which are made up of arrays of attributes. In most cases, the attributes in the same group result from running the same instrumentation code. Multiple groups can be defined within the single component definition.

An *Attribute* resembles closely the accessible entry definition in SNMP MIB, except for the value statement. The value statement indicates how and where the value of this attribute can be obtained. If the attribute is a variable, the value of the value statement is the program to be executed (i.e., the result of the execution of the program). Otherwise the value of the value statement is a constant value for the attribute.

The MIF file of the optical crossconnect switch is presented in Appendix A.1.

The procedure for using the DPI/DMI framework to develop and operate a subagent is as follows:

1. Define the MIF and create an entry for this component in the mapping file.
2. Convert the MIF file to SNMP MIB and load the MIB file to the SNMP manager side.
3. Find out the registration number for this component (i.e., the registration number of the last registered component plus 1), the number of the group, and the number of instances for this group. These numbers are needed for the implementation.
4. Write C/C++ code (instrumentation) according to the MIF and compile it.
5. Register this component to the layer service.
6. Run files *sva\dm\bin\snmpd.exe*, *extagent.exe* and *dmisa32.exe* to activate the master agent, NT MIB-2 agent (as a DPI subagent), and DMI subagent, respectively.
7. Browse the values of the variables of the component using the MIF browser on the local computer.
8. Use an SNMP manager on any remote computer to make SNMP queries.

During the course of the project, a DMI SNMP subagent framework was used to implement an agent for the optical crossconnect switch as presented in the next subsection.

2.2.4 Optical switch agent

An optical crossconnect switch is one of the critical devices in the optical ATM networks. A DPI/DMI SNMP subagent has been developed for the switch on a NT3.51 workstation (Fig. 2.3). The workstation is connected via a RS232 serial port to the microcontroller MC68HC119E which is compatible with AS11 Assembler and is used to run the control logic of the optical switch. Since the DMI API was written in Visual C++ 2.0, the instrumentation code was also written in the same programming environment. The functions

that correspond to SNMP *get*, *getnext*, and *set* were implemented in the subagent's components and tested successfully.

This implementation is a typical proxy agent. The microcontroller does not support the necessary network protocol suite TCP(UDP)/IP/UNP and thus can not form an independent network host. Therefore the functions that an SNMP agent performs have to be implemented on a separate host on behalf of the optical switch instead of on the microcontroller itself. Of course, the proxy agent can be an independent SNMP agent or a subagent. A proxy agent is very suitable for small and experimental devices.

The switch is used to switch the wavelengths between the fibers. In addition to the port connections, the current for each gate and the temperature for each port are important parameters to be managed. While the whole MIF definition file is presented in Appendix A.1, the remaining part of the subsection briefly describes the implementation experiences of the switch.

The subagent was implemented as a *Direct-interface* program so that the switch process runs continuously. This gives a faster response than an *Overlay* implementation. However, if use of the CPU is expensive, the *Overlay* method should be employed.

The MIF file contains two groups. The first group contains the definitions of the static data of the switch, which include the definition of the data type, identifications of the software, installation information, communication port and channel, and the number of the switch ports. These parameters remain unchanged once the component is activated. The second group contains the dynamic parameters which include the status, power, temperature, and current for each port, broadcast and pass-through status, and finally the connections.

This agent was tested successfully. It has proved that the DPI/DMI agent kit is a suitable tool for developing SNMP agents and subagents with high extensibility and ease of implementation and instrumentation.

2.3 SNMP Manager

An SNMP system follows in the client/server model. The agent plays the role of the server and the protocol manager plays the role of the client. While the previous sections answer the question of how to implement an agent and how the agent works, this section presents an SNMP protocol manager software, the CMU/UCD package. With this software, the user can write applications on top of the manager software. The protocol manager takes care of retrieval of the data from the SNMP agents. The application programmer just pays attention to the applications themselves.

2.3.1 CMU/UCD package

Carnegie Mellon University (CMU) provides in the public domain an SNMP implementation package that is written in C.^[15] However it is hard to port it to other platforms. In the meantime, the University of California at Davis (UCD) fixed some bugs of the CMU package and added some scripts to check the availability of the libraries needed on the computer where the software is installed. The current version is ucd-snmp-3.1 and the package can be downloaded from Website free of charge.^[16] This package is highly portable and was successfully installed and tested on Sparc/SunOS4.1.3 and Solaris2.5, and DEC/OSF3.2 systems.

The CMU/UCD code supports SNMP version 1 and 2. The implemented features include, *snmpget*, *snmpgetnext*, *snmpwalk*, and *snmpset*. The *snmpget* allows the user to retrieve the value of a given instance. The *snmpgetnext* gives the value of the object instance next to the specified object that does not have to be instantiatable. The *snmpwalk* allows the user to retrieve the values of all instances of a group of variables from the agent without having to specify the exact instances. The *snmpset* sets the value of a single instance. All these functions were successfully tested on three platforms. In addition, there is also a *trap* facility for the agent to send some critical messages to the manager. The module *snmpctest* provides an API for the user to include the SNMP facilities in the high level applications.

2.3.2 MIB tree and query examples

The NM information provided by the agents to the managers is logically stored in the MIB. Fig2.4 shows the subtree which is relevant to this project. This is a small portion of the whole MIB tree and resides beside the CMU/UCD SNMP manager package.

In this node tree, every intermediate node represents a group of objects. Every leaf node stands for a type of object. Very often “object” alone is too abstract to represent the exact meaning in the context of SNMP so that I prefer calling a leaf node in the MIB tree “a type of object” instead of just “object”, although many authors do use “object” in the same context. From the type of object, instances (also called object instances) can be produced. It is the object instance that holds the values of the object.

To reference a value of an object instance, the path in the MIB tree and identifier of the object instance have to be specified. For example, to retrieve the maximum transfer unit, MTU, i.e., the maximum number of bytes in a packet for a given network interface, the following query should be used

Figure 2.4 The interesting portion of the MIB tree.

```
snmpget -v 1 alpha public .1.3.6.1.2.1.2.2.1.4.1
```

where *snmpget* is the *get* function of the CMU SNMP package, *-v 1* means version 1, *alpha* is the name of the host where the agent resides, *public* is the community name, and the string of the digits and dots is the identifier for the Ethernet card for this workstation.

Usually, the complete MIB file which the agent supports resides on the user host as the manager. If this is the case, a query includes the requested value along with the full variable name; i.e., the name of variable is resolved. For example, the query mentioned above gives

```
interfaces.ifTable.ifEntry.ifMtu.1 = 1500.
```

However, if there is no such MIB file on the manager side, the variable's name is not resolved although the value itself may be correct. Thus the result of the same query would be

```
.1.3.6.1.2.1.2.2.1.4.1 = 1500.
```

In Fig.2.4, the branch starting at *mgmt (2)* is the standard MIB-II. All workstations involved in the project support it. The branch starting at *Carleton-NRC(99)* stands for the agent of the optical switch that has been implemented and only machine *rho* supports it. *fore(326)* is the starting node for all MIBs designed in FORE ATM switches.

This chapter can be summarized as follows. A TMN-like functional hierarchy is preferred for use in the NMS of the optical ATM networks. The DPI/DMI APIs provide a suitable tool for writing SNMP subagents. A DMI subagent for the optical crossconnect switch was implemented and tested successfully. The UCD SNMP manager package has proven to be a good starting point for developing SNMP-based management applications.

Chapter 3

MANAGEMENT COMMUNICATIONS

As discussed in Chapter 2, the TMN-like functional hierarchy is a better choice for the design of a NMS for optical ATM networks. Because the quality of the communications of the network operations and the management messages have a significant impact on the performance of the NMS, it is worthwhile to make a comparison of the performance of the client/server systems that are written in the different network APIs available. This chapter discusses the network APIs, client/server systems, and their performance. §3.1.1 introduces the Internet domain protocol hierarchy and the basic client/server model. §3.1.2 briefly describes the main features of the BSD sockets and a TCP client/server system. §3.1.3 discusses the FORE System's ATM-API and a corresponding client/server system. §3.2 presents the experimental setup and the results of the performance analysis of the client/server systems written in the two APIs.

3.1 Network APIs

3.1.1 Client/server model

In a NMS, communications between different NM entities are carried out via standard protocols. From a user's point of view, data is transported using various client/server systems that are implemented with relevant network APIs. In the Internet world, the dominant network API is composed of the BSD's sockets. This API is an interface between the transport layer (TCP/UDP) and the application layer. Most network applications on various UNIX and Windows are implemented with sockets. Sockets are widely used both for primitive client/server communications and for sophisticated systems such as SUN RPCs, DCE RPC. An ATM-API, on the other hand, is an interface between an ATM adapter and the user. The ATM-API allows two ATM hosts to communicate directly through the ATM network without TCP/UDP support. The network protocols and the APIs hierarchy are shown in Fig.3.1. In a typical Internet and ATM hybrid network environment, each host is connected to an Ethernet and an ATM network. In such a case, two APIs and three paths are available for communications to take place:

- Sockets-TCP/UDP-IP-UNP
- Sockets-TCP/UDP-IP-AAL-ATM-UNP
- ATM-API-AAL-ATM-UNP.

The socket interfaces provide one more degree of reliability by using TCP/IP services for ensuring data integrity. The very obvious advantage is the application transparency in that the same program can run on different machines and different networks. The ATM-API, on the other hand, provides a socket-like interface for the user to write applications directly

running on the ATM network without passing through the TCP/UDP-IP stack. This approach leaves the end-to-end flow control between hosts and cell retransmissions to each application. The following subsections describe the client/server systems written in the two APIs.

Figure 3.1 Network protocols and APIs hierarchy.

3.1.2 BSD Socket

The communications path on the Internet is application-UDP/TCP-IP-UNP. BSD sockets provide an interface between transport layer and the application. The TCP client/server system is illustrated in Fig.3.2. Here *fd* is really a file descriptor for a particular socket.

Figure 3.2 TCP/IP socket client/server model.

To deal with multiple clients, the server always uses the initial socket to listen for new connections. Upon receiving a new connection request, the server produces a new socket exclusively for the new connection. The new process manages the new connection so that the original socket keeps listening for new connection requests. Thus the server can handle multiple clients concurrently. Our performance experiments measured the round trip latency. To do so, a client sent messages of variable length to a server, which upon receipt, immediately returned each message. All messages were transferred on duplex connections. The system call *gettimeofday()* was used to measure the latency and *getrusage()* was used to measure the CPU time. The TCP/IP client/server system program was tested successful in such platforms as DEC/OSF3.5, SGI/IRIX5.3, and Sun/Solaris2.x.

It should be pointed out that the same client/server programs can run on TCP-IP or TCP-IP-ATM stack without any modifications. The Ethernet card and ATM card use a different host name and an different IP address. Suppose that the server is on the machine *alpha* for the Internet and *alpha-atm* for ATM, and the client is on machine *omega*. The following command lines are used to run the client application on Internet and ATM network respectively:

```
omega> myclient alpha
omega> myclient alpha-atm.
```

Besides, the client/server programs can also be written using UDP in a slightly different way than using TCP.

3.1.3 ATM API

FORE Systems' ATM cards come delivered with an ATM-API that does not need TCP/UDP support. The functions of the ATM-API are similar to that of BSD sockets. One of the advantages of the ATM-API is the negotiation of the QOS as shown in Fig.3.3.

The QOS can be specified in target mean and minimum mean bandwidth, target mean and minimum burst data length, AAL, and MTU. In doing so, first the user specifies the required QOS parameters in the client program and these parameters are sent to the server. After receiving the message from the client, the server specifies its available QOS and sends the information to the network for negotiation. The final QOS parameters are returned to the client through the function *atm_connect()* and the negotiation is completed. However, there is no flow control in the ATM-API and error checking is left to the application.

Figure 3.3 ATM-API model

3.2 Client/Server Performance

As discussed previously, in an Ethernet and ATM hybrid network environments there are three candidates for use as a management network: Sockets-Internet, Sockets-ATM, and ATM-API-ATM. For optical ATM networks, a few Mbps out of several Gbps bandwidth may be assigned for the purpose of NM at a low cost. The choice of a management communication channel mainly depends on the performance, especially in terms of speed and reliability. Thus it is worthwhile to make a comparison of the performance on the three client/server systems.

3.2.1 Experimental setup

Fig. 3.4 shows the experimental setup for the client/server performance measurements. For all measurements, the two hosts in a client/server system were in the same LAN (Ethernet) and connected to the same FORE ASX200WG ATM switch.

Figure 3.4 Experimental setup for client/server performance measurements.

Three characteristics of the APIs were measured: round trip latency, throughput, and jitters. Jitters are defined as the standard deviations of the latency and were calculated with 95% confidence. For each message size, 100 iterations were conducted in order to obtain an average latency and bandwidth and the jitters.

For purpose of control and management, a message of 1k bytes should be long enough, although for applications such as multimedia communications, much longer messages will be

needed. Therefore the length of the experimental messages extends from a few bytes to 25 kbytes.

The following two subsections present the results of the measurements for two study cases. §3.2.2 shows the measurements for Internet, and for TCP/ATM on machines *ir2* and *spin191* as shown in Fig.3.4. These two machines were located in the AI ethernet. The data behave normally. §3.2.3 shows the results for Sockets and ATM-APIs running on *autreal* and *indy*. These two machines were located in the SGI ethernet. The data for socket-related data behave abnormally and explanations are given there.

3.2.2 Case 1: Normal behavior

Fig.3.5 shows the round trip latency for TCP/Ethernet client/server system running on the ethernet. The two machines *ir2* and *spin191* are Sparc/Solaris2.5. The CPU time was measured on the client side so that the curve $2 \times \text{CPU}$ is approximately equal to the whole system's CPU portion of the round trip latency. At about 17 kbytes, the latency increases very rapidly. A possible reason for this is the buffer size. One buffer can only accommodate this amount of data. The large part of the latency is unlikely attributed to the transmission and propagation time of the network. In a LAN, the data transmission time of a 10 kbytes message with 10 Mbps bandwidth is $10^4/10^7=1(\text{ms})$. The propagation time for an 100m LAN is about: $100/(2 \times 10^8) \approx 0.0005\text{ms}$. The possible reasons for the gap between the overall latency and the overall CPU time are having an inefficient Ethernet driver and MAC contention. In the latter case, the Ethernet bus may be busy or requests messages for media access frequently

encounter collisions with other requests. The two problems may be caused by sending and receiving the experimental data or by other users' traffic.

Figure 3.5 Round trip latency and CPU for TCP/IP/Ethernet.

Fig.3.6 shows the latency for TCP/AAL5/ATM communications. The client/server system and the configuration of the network are exactly the same as in Fig.3.5. The introduction of the ATM technology to the computing system largely reduces the latency and thus improves the network performance significantly. The latency for TCP/AAL5/ATM communications is about one fifteenth of the value for TCP/Ethernet communications. Fig.3.6 clearly indicates that the latency is mainly determined by the CPU time. A faster CPU machine would reduce the latency further. The small steps at multiples of about 9100 bytes are caused by the segmentation of the ATM packets. One protocol data unit (PDU) of ATM is

equal to 9188 bytes. The maximum amount of data that can be transferred in one packet is about 9100 bytes because of TCP and IP headers (a total of 40 bytes as specified in files `/usr/include/netinet/tcp.h` and `ip.h`) and other overhead from the ATM sublayers. Unfortunately, ATM-API could not be tested on these two machines at the writing of the thesis due to some configuration problems.

Figure 3.6 Round trip latency and CPU for TCP/IP/ATM.

3.2.3 Case 2: Abnormal behavior

Fig.3.7 shows the round trip latency for another TCP/IP/Ethernet client/server experiment. The data were measured by running the same program as in case 1, but on different machines (SGI/IRIX5.3) and in a different subnetwork, i.e., *autreal* and *indy* in

the SGI ethernet. The latency spectrum has peaks with regular shape and all the peaks are separated in a equal distance. A close look shows that except the first one, all the peaks start at $(n+1)*1460$ bytes, and end at $n*1460$ bytes where n is an even integer. These peaks are directly related to the network MTU and the TCP sliding window, as explained below.

Figure 3.7 Round trip latency of TCP/IP/Ethernet.

With sockets, any data to be transferred to the network has to be packed in a packet called a Network Protocol Data Unit (N-PDU). The MTU of the underlying network was found to be 1500 bytes with a 20 bytes TCP header and a 20 bytes IP header, and the maximum length of user data in one N-PDU is $1500-40=1460$ bytes long. This explains why the edges of the peaks are at multiples of 1460 bytes.

The reason why the peaks alternate regularly is due to the TCP sliding window.^[17] When the length of the user data L is less than one MTU, the sender sends the packet immediately without any delay. When $MTU < L < 2*MTU$, the sender sends the full segment first and does not send the remaining part immediately. The sender tries to gather more data from the application while waiting for an ACK from the receiver, in an attempt to make a full packet. At the same time, the receiver is waiting for more data from the sender for completion of the whole message without sending an ACK for the first packet. Two hundred milliseconds after the moment when the receiver received the first packet, the receiver sends back an ACK. When the sender receives the ACK, it sends out the remaining part of the data. Thus the 200 ms time-out is the main source of the latency delay. When $2*MTU < L < 3*MTU$, the sender sends the first two full packets and get an ACK from the receiver, then sends the remaining part out right way without any delay.

The intention of this mechanism is to improve the network traffic efficiency. If the sender could gather more data while waiting for ACK, then the efficiency of the network transfer would be higher. A socket option `TCP_NODELAY` is designed to control this mechanism and the default value is disabled. This option can be enabled by invoking function `setsockopt()`. During the course of the research much effort was paid to turn on this option. Unfortunately it was not successful. Another way to turn around this 200 ms time-out problem is to pad the remaining segment manually in the application and send it immediately after sending out the full segments. This is simple and practical approach and was used in the following measurements. Similar result was obtained for TCP/IP/ATM.

Fig.3.8 shows the latency distributions for the three experiments: TCP/IP/Ethernet, TCP/IP/ATM, and ATM-API. All fractional parts of the messages were padded to a full

packet. The latency of the Ethernet increases rapidly at about 17 kbytes. The latencies for TCP/IP/ATM and ATM-API are close to each other although the former has bigger fluctuations due mainly to the network access contention in the underlying Ethernet.

Figure 3.8 Comparison of the round trip latencies for ATM-API, TCP/IP/Ethernet and TCP/IP/ATM with unfull packets padded to full.

Fig.3.9 contains the same results of the same experiment run for messages less than 1 kbytes. The latency for Internet increased at a rate $5\mu\text{s}/\text{byte}$ while the other two variables are almost constant. ATM-API incurs a latency of about two-thirds of that of TCP/IP/ATM, and in a more predictable way.

Figure 3.9 Comparison of the round trip latencies for ATM-API, TCP/IP/Ethernet and TCP/IP/ATM with unfull packets padded to full for messages < 1 Kbytes.

Fig.3.10 shows the throughput distribution of the three cases. The ATM-API gives smaller fluctuations than do TCP/IP/Ethernet and TCP/IP/ATM. For TCP/IP/ATM, a large inverse-triangle-shaped dip exists at around 10 kbytes, which is undesirable for real-time applications. However, for ATM-API, a message around 3 kbytes can achieve the maximum bandwidth. For TCP/IP/Ethernet, the throughput reaches its maximum 6 Mbps at around 12 kbytes and then drops to about 3 Mbps. This is mainly due to the network access contention.

The experiments clearly show that ATM-API has some advantages and some disadvantages over TCP/IP/ATM. The ATM-API does not have sliding window control. When using the ATM-API, the user does not have to worry about the problem of the 200ms time-out as is the case in TCP/IP. Since the communications take place in a dedicated ATM path, the fluctuations are much smaller than that in TCP/IP applications. Thus using ATM-API makes the latency more predictable. However, the user has to take care of the

segmentation. If the user data is longer than one segment, it has to be segmented to two or more segments manually in the program. Another problem is that there is no flow control in the ATM-API. Some mechanism, like SSCOP (Service Specific Connection Oriented Protocol), has to be implemented. More information about SSCOP can be found from the literature.^[18,19]

Figure 3.10 Comparison of throughput of the ATM-API, and TCP/IP/ATM and TCP/IP/Ethernet with unfull packets padded.

In summary, this chapter shows that the order of increase in latency and the jitters are: ATM-API, TCP/IP/ATM, and TCP/IP/Ethernet. While the latency of the TCP/IP/ATM is

close to that of ATM-API, the fluctuations are much larger. Maximum bandwidth can be reached at about 3 kbytes and 17 kbytes for the ATM-API and TCP/IP/ATM, respectively. The throughput for TCP/IP/Ethernet becomes worse for messages larger than 15 kbytes. In consideration of reliability, the TCP/IP/ATM is a better candidate for use as the management network in a large NMS. However, if the management messages are less than MTU or some techniques like SSCOP is implemented, then ATM-API is a better choice.

Chapter 4

OPTICAL ATM NETWORKS

To meet the increasing demand for high bandwidth and traffic complexity, the telecommunication standards bodies have recommended ATM as a transport, multiplexing, and switching technique for Broadband Integrated Services Digital Network (B-ISDN). ATM combines the advantages of both circuit and packet switching techniques, allowing all services to be transported and switched in a common digital format. ATM control signals are processed along with the payload, and switching decisions are made on the fly. It is difficult to speed up the processing electrically and difficult to implement ATM processing with optical components. The electronic processing is a bottleneck when the aggregate traffic is more than a few Gbps.

Fortunately, SONET has been recommended as the physical-layer transmission standard. ATM, together with SONET, is expected to provide a high-speed and reliable transport, bandwidth flexibility, and integrated transmission and switching for the diverse set of traffic characteristics required by B-ISDN. Optical networks provide enormous potential

for high speed networking. While the classical TDM approaches realized in electronic circuit are gradually becoming insufficient,^[20] various all-optical approaches are being investigated to shift the routing work load from electronic ATM switches to the optical network nodes. It is expected that optical ATM hybrid networks with hundreds of Gbps will become a reality in the near future.

Since the primary purpose of the thesis is to identify the practical design issues of the next-generation NMSs for high-speed optical ATM networks, the target network is not meant to be any particular network. Instead it is a collection of the currently available optical networking technologies within a reasonable architecture.

To build a state-of-the-art NMS, one has to face the challenges that come from the complexity of the optical ATM networks. This chapter aims at giving an overview of designs for a high speed optical ATM network and provides a solid technical background for the design of the NMS architecture proposed in Chapter 5. It begins with an introduction to ATM technologies. Then the advantages of the optical networks are described. Finally an optical ATM network architecture is discussed.

4.1 ATM Network

This section describes the basic concepts of ATM technology, ATM interfaces with SONET, and some potential problems in developing high-speed networks.

In ATM networks, data are transferred in cells of fifty three octets, five of which are header information and the remaining 48 octets are user data. Sometimes, the payload contains some operation and management information. The cell routing information contained

in the header is a label, which is the concatenation of the VPI (Virtual Path Identifier) and the VCI (Virtual Channel Identifier) fields. ATM cells are transported along the entities known as virtual Channels (VCs) that are identified by their VCIs. VCs are carried on within Virtual Paths (VPs) that are identified by their VPIs. VPs integrate VCs to provide an unstructured data pipe.

The VPI and VCI together form a 24-bit Protocol Connection Identifier (PCI). The PCI identifies a particular call and is used for routing cells across the network, multiplexing cells at the source, and demultiplexing cells at the destination. For a given connection, both VP and VC are subject to switching within the ATM network. They are local variables within the switches. A VP switch can redirect a VP, perhaps reassigning the VPI, but it keeps the VCs within the VP intact. A VC switch must terminate VPs and can switch the VCs within a VP independently of each other. All this switching means that the VPI and the VCI at different ends of a connection might not be the same.

Connection setup can be point-to-point, or point-to-multipoint. It is not yet decided whether ATM connections are unidirectional or not.^[21] If they are, then a reverse connection has to be used for full duplex connections. The reverse direction might be used for a management purpose since ATM is not implemented with acknowledgment. Connection setup can be done at subscription time or on a per-call basis.

In contrast to other existing networks where some kind of window flow control is provided, ATM layer uses rate control between the user and the network access node. Every connection is associated with a traffic contract negotiated between the user and the network. The contract includes such terms as mean bit, peak bit rate, and peak duration.^[21]

In order to carry data units longer than 48 octets in ATM cells, the ATM adaptation layer (AAL) is provided along the ATM standard. The AAL provides for segmentation and reassemble of higher-layer data units. Since the ATM layer simply carries cells without concern for their contents, a number of different AALs can be used across a single ATM interface. The end points of each connection must agree on which AAL they will use, but the network need not be aware of this. The five AALs accepted for consideration by the ITU-T are: AAL1, 2, 3, 4, 5. Table 4.1 lists the different services supported by the AALs. Detailed descriptions and comparisons of the services can be found in the literature.^[22,23]

Table 4.1 ATM services

AAL	AAL1	AAL2	AAL3/4		AAL5	
Class	A	B	C	D	C	D
Timing*	Required		Not required			
Bit rate	CBR	VBR				
Connection mode	Connection oriented			Connection less	Connection oriented	Connection less
Multiplexing	No	No	Yes		No	
Characteristics and data Examples	voice	audio and video	data which is sensitive to loss, but not to delay		efficient signaling and data (e.g., Frame relay, SMDS, MPEG-2 video)	

* timing between source and destination

B-ISDN standards support the transmission of data using both synchronous transfer mode (STM) such as SONET and ATM for layer 1 and layer 2, respectively.^[24] The ultimate

goal is to transfer data throughout the entire network using the ATM switching technique for the end users and using STM as the high bandwidth transport network protocol.

A single mode optical fiber can operate at approximately 25 THz, corresponding to a data rate higher than 1Tbps.^[24] The current operational bandwidth in ATM is only a few Gbps which is far below the potential of the fiber's intrinsic capability. ATM cells are switched independently of each other and based on control information in the cell header. Control signals are processed along with the payload and the switching decisions are made on the fly. This makes it difficult to speed up the processing electrically and difficult to implement ATM processing with optical components. To provide high bandwidth capacity, one has to use multiple electronic channels to process ATM signals for an aggregated high data rate. However this can become very complex in a large scale communications networks.

On the other hand, the switching pattern for STM signal is predetermined and held constant for an interval of time. Consequently, STM control signals can be separated from the data. With STM, the length of one packet (9188 bytes for OC-3) is much larger than the ATM cell (53 bytes). These factors make STM processing of high speed data considerably simpler than ATM processing.

Therefore, a new B-ISDN architecture for high bandwidth is desirable. In this architecture, STM and ATM technologies should be combined together in order to bundle multiple ATM connections and send the data to the STM. The above reasoning implies that the key component in this architecture is the optical multiplexer. There may be many different approaches to take advantages of both ATM and optical networks. The decision is made based on such factors as the network development strategy, the size of the network, and the distribution of the network facilities.

4.2 Optical Transport Network

The predicted growth in demand for broadband services has led to extensive research into high capacity network systems. It is predicted that within two decades the transport layers of the network backbones may need to support capacities 100 times greater than that at present.^[25] The currently deployed transmission technology SONET and the switching technologies such as ATM will meet the near-future-term needs of the network infrastructure. However such approaches are based on electronic signal processing and thus network solutions based on these approaches will become increasingly complex and expensive. Therefore it is believed that optics will play an ever increasing role in the future network implementations.

The capacity of a fiber is far greater than what is currently used (by at least a factor of 1000).^[25] This potential capacity can be realized in two possible ways: WDM and TDM. The whole idea in both approaches is to use the available fiber spectra efficiently. The first approach is to multiplex signals on different carrier frequencies (wavelengths) onto a single fiber. The second approach is to use extremely narrow pulses to produce high bandwidths by using optical TDM. In the future, networks may incorporate both WDM and TDM.

Another important optical technology is optical switching and routing. Currently almost all network signals are processed electronically. For example, in the upper layers on the SONET network, a cross-connection within a 2.5 Gbps multiplex is achieved electronically. It will become more and more difficult to implement electronic ATM switches in an efficient and cost-effective way as higher and higher bandwidth capability is required. The optical switching and routing technology helps reduce the demands on electronic cross-connects. For

large networks, the wavelength routing enables paths to be established through a network and uses wavelength to define the route.^[25]

4.2.1 SONET

SONET is a standard for optical TDM systems proposed by Bellcore and was adopted by ITU-T in 1989.^[22] SDH (Synchronous Digital Hierarchy) is very similar to and differs only in minor ways from SONET. While the former is mainly used in Europe, the latter is used in North America. The goal is to define a common signaling standard with respect to wavelength, timing, and frame structure so that different carriers can network.

Fig.4.1 shows a SONET path. A SONET system consists of three kinds of devices; switches, multiplexers, and repeaters that are all connected by fibers. A fiber between any two devices is called a section. A run between two multiplexers is called a line. A connection between the source and the destination is called a path.

Figure 4.1 SONET path.

Fig.4.2 shows the structure of the SONET frame layout. The basic SONET frame is a block of 810 bytes (90 columns by 9 rows) emitted every 125 μ sec. Since SONET is synchronous, frames are transferred does not matter whether there are any useful data to send or not. This speed (8000 frames/sec) exactly matches the sampling rate of the PCM (Pulse Code Modulation) channels at 64 kbps $[(8 \text{ bits/sample}) \cdot (8000 \text{ samples/sec})]$ used in all digital telephony systems. That is, one of the 810 bytes carries a PCM channel.

Figure 4.2 SONET format.

As shown in Fig. 4.2, the first 3 columns of each frame are reserved for system management information (path, line, and section), and the remaining 87 columns hold $8 \cdot 9 \cdot 87 \cdot 800 = 50.112$ Mbps of user data, called SPE (Synchronous Payload Envelope). The

gross data rate in this layout is $8 \times 810 \times 8000 = 51.84$ Mbps. This basic SONET channel is called STS-1 (Synchronous Transport Signal-1) for electronic processing and OC-1 for optical processing. All SONET trunks are based on this basic channel.

The multiplexing of multiple data streams, called tributaries, plays an important role in SONET. Fig. 4.3 shows an example of multiplexing hierarchies from phone lines to an OC-12 trunk. Multiplexing is done byte by byte. For example, when three STS-1 tributaries are merged into one STS-3 stream, the multiplexer first outputs one byte from tributary 1, then one from tributary 2, and finally one from tributary 3, then goes back to stream 1. The result is an output stream at 155.52 Mbps that is STS-3 for electrical connections or OC-3 for optical connections. In case the 155.52 Mbps stream carries data from a single source, the SONET should be denoted by OC-3c.

Figure 4.3 An example of multiplexing hierarchy in SONET.

Table 4.2 lists the often-used multiplexing rates.^[22] In the table, the user data means the data from the end user's application whereas the SPE means the payload of the SONET trunk which includes some data as multiplexing information.

Table 4.2 SONET and SDH multiplexing rates.

SONET		SDH	Data rate (Mbps)		
Electrical	Optical	Optical	Gross	SPE	User
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC-3	STM-1	155.52	150.336	148.608
STS-12	OC-12	STM-4	662.08	601.344	594.432
STS-24	OC-24	STM-8	1244.16	1202.688	1188.864
STS-48	OC-48	STM-16	2488.32	2405.376	2377.728
STS-192	OC-192	STM-64	9953.28	9621.504	9510.912

4.2.2 Optical networking

Advancements in optical components and system technologies have made it possible to develop Gbps-level optical transport networks.^[26-29] Basically, two different but complementary transport architectures can be implemented to meet the scalability requirement. The first one uses a pure ATM cell time multiplexing on 1.25, 2.5, or 9 Gbps fiber rings. These bitrates correspond to the total capacity of the two rings (bidirectional) for OC-24, OC-48, and OC-192 SONET. The second one uses WDM in a multihop network, which carries SONET frames. The following paragraphs describe the main features of the two approaches.

In both cases, a primary concern is to route the traffic away from the electronic ATM switches that are used as the users' access nodes to the high-speed network. The ATM switches are then only loaded by the traffic that is originating from or destined to a site covered by the switch. This idea allows the lowest possible end-to-end transfer delay.

In a double ring structure, the optical fiber rings interconnect the ATM switches through the dedicated access nodes (Fig. 4.4). The access nodes perform ADM function that are capable of routing the ATM traffic based on the ATM header value on a cell-by-cell basis, independently of the ATM switches. The direct transfer capability minimizes the transfer delay, which can be kept below $10\mu\text{s}$ for a 1.25 Gbps cell stream.^[30]

Figure 4.4 Double ring structure of optical ATM network where an X stands for one or more ATM switches.

The ADM nodes have a high-speed optical front-end to receive and demultiplex the incoming ATM traffic.^[30] The demultiplexed traffic is forwarded to a completely integrated processing core. The core is organized in a bit slice architecture and performs all physical and

ATM layer functions such as scrambling, cell delineation and multiplexing, and address filtering. Based on the address value in the ATM cell header, a decision is taken as to either transfer the cell to the next ADM node or drop it to the local switch. Cells can be inserted from the switch into the optical backbone through the access node.

The double ring is a particularly robust architecture. In case of link failure, the traffic is routed directly at the optical level using fast optical protection switching.^[30] The cell processor is able to recover the cell synchronization in a one step process which guarantees that at most a few ATM cells per logical link are lost in case of a fiber cut failure. As in all other ring networks, a “garbage collection” mechanism should be provided in the double ring structure in order to remove wrongly addressed cells that would circulate indefinitely around the ring otherwise.

The multi-hop optical network architecture combines SONET with a multi-hop WDM technology. As with the double ring architecture, the SONET frames are used to transport the ATM cells. In doing so, an ADM node can be used as the access node for ATM switches. The optical interconnection in large ATM networks can be fully exploited by WDM techniques. A higher capacity, more flexible, and resilient architecture can be constructed by adding suitable components at the edges of the installed fiber network.

In the double ring, all ATM cells travel along the same fiber path and with the same wavelength. In contrast, the key feature of the multi-hop WDM optical network is that ATM cells make multiple hops through intermediate optical nodes in order to reach their destinations. This reduces the load on the switches because data traffic not destined for a given node never reaches its switch's electronics, and is instead routed optically to the next

node. The network resilience is improved through the ability to offer alternative data traffic routes.

With this multi-hop optical network architecture, the fiber between two intermediate nodes may carry traffic in many different wavelengths. As an ATM cell travels from the source to the destination, it may undergo different wavelengths. Also a single wavelength can be used in different paths as long as the two paths do not overlap.

When actually implementing high-speed networks, the two architectures described above can be merged to form a high capacity and flexible optical ATM networks. In the new architecture, a multihop optical network is used as a public network backbone to transport and route the high-speed traffic using different wavelengths. An WDM node is used as a transition point for the backbone network and the local SONET double ring network. The next section describes the architecture in detail.

4.3 Optical ATM Network Architecture

A schematic of the proposed optical ATM network architecture is shown in Fig.4.5. The network consists of three different functional nodes: optical transport node, optical access node, and the switch node. In some cases, the first two can be implemented in one physical node.

The optical transport node performs the functions of setting and distributing the optical paths between arbitrary access nodes, which include cross-connect and ADM. The transport node systems are connected by the optical transmission system with optical amplifiers. For

example, a point-to-point optical path such as D-E-F in Fig.4.5 can be realized without optical/electrical conversion.

Figure 4.5 Architecture of high-speed optical ATM networks.

In the network, each access node has an ADM module to add or drop the traffic between the SONET double ring and the optical network. Each node also incorporates an optical cross-connect switch so that the overall interconnection of the nodes can be reconfigured to support particular network requirements.^[25] Signals are routed through the network according to wavelength they use (wavelength routing) and their point of access into the network.

A major feature of such networks is the reuse of a wavelength. For example, λ_3 can be used to route F-E-D and B-E since the two wavelengths (λ_3 in the two paths) do not pass through the same fiber. Within each node, a combination of wavelength-selective elements together with optical space switching can assign a wavelength on any incoming fiber to any outgoing fiber. In addition, wavelength demultiplexing and space switching (the cross-connect) allow an input wavelength to be routed to any outgoing fiber or dropped off to the local SONET double ring.

Fig.4.6 shows the architecture of the optical network node located in E of Fig.4.5. Each of the three fibers connected to the node may support four wavelengths. Each wavelength in a given fiber can be routed to any outgoing fiber. Traffic within each wavelength of a fiber can be dropped to the local path via the electronic cross-connect switch and the traffic from the local network can be added to any of the outgoing wavelengths on any fiber. The electronic cross-connect process is used to multiplex/demultiplex local traffic.

Several issues deserve more explanation. Within each node, not all of the three fibers carry all the four wavelengths; instead the four wavelengths are configured according to the networking policies. An SNMP agent may be built for each optical network node to control and monitor the operations of the wavelength routing and wavelength reconfigurations. In addition, signal amplification is necessary in order to compensate the loss occurred within the

node. This can be done using Erbium doped fiber amplifiers (EDFAs). Finally, the number of useful wavelengths is limited, putting a constraint on constructing large networks that can be partially overcome by means of wavelength reuse and conversion.^[25]

Figure 4.6 Optical network node.

Various strategies can be applied to the design of an optical network. For example, one can reserve one of the available wavelengths to carry ATM traffic for network management as

TMN does. Figs.4.5 and 4.6 show this configuration as an example. While some technical problems still need to be carefully studied and the cost-effectiveness need to be addressed, this architecture takes advantages of the ATM and optical networks and shows the unique potential of the high-speed optical networks.

Chapter 5

MANAGEMENT ARCHITECTURE

As discussed in the previous chapters, ultra high-speed networks consist of optical networks and ATM switches. Any NMS for large networks must be able to work in a heterogeneous environment. To deal with these requirements, a powerful and well designed NMS is needed. This chapter begins with a brief description of the fundamentals of the NMS. Then a TMN-based framework for the NMS is proposed. The chapter concludes with discussions of various aspects of the architecture of the next generation NMS that include interface modules, network managers, and a GUI.

5.1 Basics

There are two categories of messages in network systems. The first category is the messages controlled by the NMS. The purpose of these messages is to accomplish the normal network operations and maintenance (OAM) and NM messages. The second category is the user traffic. To accommodate the two kinds of messages, network systems are organized into one of three

modes. The modes are in-band-in-fiber, out-band-in-fiber, and out-band-out-fiber, as explained below.

In the in-band-in-fiber mode, the traffic is organized in a specific format where the header and a small portion of the payload is for the OAM and the major part is for user data and NM. In this configuration, a great deal of work on hardware and software has to be done in order to recognize the different messages. The out-band-in-fiber mode can be employed in optical networks. One wavelength is specified for NMS and the others for user traffic. One can also use one wavelength but in amplitude modulation where the high frequency is for user data and the low frequency for the OAM and NM messages. In out-band-out-fiber mode, two physically independent networks are involved; one is for user traffic and the other for NM.

As mentioned in the previous chapters, it is better to follow the TMN framework in building up the functional hierarchy for the NMS. In doing so, the functions of the TMN nodes and interfaces, as defined in the TMN document,^[11] do not have to be fully implemented. The two logical networks in the TMN standard can be in either out-band-in-fiber or out-band-out-fiber mode, depending on the existing optical networks. Every node or host in the system is logically connected to the two networks. One is the TMN managing network and the other one is the managed network. While the managed network transmits the high bitrate traffic such as multimedia, the managing network carries the OAM and NM messages.

Since ATM networks are being deployed widely, an ATM network is assumed to play the role of a managing network for the architecture of the NMS. For the optical network depicted in Figs.4.5 and 4.6, a specific wavelength λ_0 spins the whole optical network and

reaches each and every node. This wavelength carries the OAM and NM traffic. In fact, this architecture falls into the out-band-in-fiber mode.

This is the hardware, or more precisely, network part of the architecture for the NMS. The next section presents the software components of the architecture.

5.2 Management Framework

There are two basic NM protocols that are widely accepted, namely SNMP and CMIP. In telecommunications, the TMN is more and more adopted as a standard for the architecture of the NMS. TMN supports CMIP and makes use of its set of services CMISE. In addition, many switch providers have their proprietary management protocols built in to the network. SNMP is most often used in data communications and the CMIP-based TMN is used in the telecommunications.

This boundary definition of the SNMP and CMIP-based TMN made a sense before ATM became a reality. At that time, data communication networks such as the Internet and telecommunication networks such as telephony networks were separate from each other. Data communications were relatively simple and did not need a high speed NMS. Telecommunications were complex systems and thus needed a powerful and complicated management system. The object-based TMN architecture met this requirement. It was easy to determine which system (SNMP or CMIP-based TMN) should be implemented for a specific network.

However this view is no longer valid due to the wide scale implementations of ATM networks. ATM is designed to integrate data, voice and video to a single network. Some telephony companies have already started to transfer their data and telephony services via a single ATM network. Therefore using an ATM network to carry OAM and NM messages guarantees the quality of data transmission and also can help upgrade ATM networks to optical ATM networks.

Heterogeneous environment is another factor that has to be taken into account in a NMS. Various pieces of network equipment come from different vendors and may support different NM protocols. In case of the NRC network, one of the two ATM switches was provided by FORE Systems and supports an SNMP agent. The other came from Newbridge and supports its proprietary management protocol. An NMS must be implemented in a layered architecture in order to deal with a heterogeneous environment.

Fig.5.1 shows the framework of the proposed architecture for managing the optical ATM networks. The architecture adopts the TMN's layered functional structure. A complete layer stack consists of the EL, EML, NML, and SML as defined in Chapter 2. A brief description of the system is given in this section and detailed discussions are presented in the following sections.

The EL consists of the protocol agents that are defined for the physical devices. An agent is a piece of software residing either on the physical device or as a proxy agent on a secondary host computer that has a specific communication channel with the managed device. For example, when a switch is controlled by a microcomputer as is the case in the project, a piece of software has to be installed on a computer (here on NT3.51) as a proxy agent to monitor

Figure 5.1 Framework of corporate ATM optical networks management system.

and control the switch. The term element means the entire entity of the device: the physical device and the software that are in charge of communications. Some aspects on how to implement SNMP agent have been presented in Chapter 2.

The EML is generally used to manage a group of devices that perform the same functions. In the physical network structure presented in Chapter 4, several ADMs, for example, are connected to a SONET ring. They cooperate to carry out the same task for the ring synchronization and multiplexing/demultiplexing so that these ADMs should be treated as one entity in the EML.

The NML is where the network managers reside. The managers communicate with the agents either directly or via the EML when applicable. In either case, the communications are carried out via interface modules. Usually the software of a protocol manager resides in the interface modules. When a network manager requests data, it sends the request to the interface module first. The corresponding protocol manager residing in the interface retrieves the data from the agents using specified NM protocols. The interface modules translate the data from a protocol-dependent format into an internal format and forward the data to the network manager. Immediately above these protocol managers, there may be some interface gateways that serve such purposes as access admission control of the data use, a directory service for the upper layer telling the network managers where and how to get the data, i.e., which modules should be invoked. These pieces of software should reside on the computers where the network managers are, in order to reduce response delay of the retrieval process. §5.3 describes some issues in implementation of the interfaces.

In the NML, three network managers may be plenty enough (for resource management. These managers are for configuration, fault, and performance management. §5.4 presents some details on the functions that the network managers perform.

In the SML, an end user access the network management services via a remote access. The access mechanisms include home-made client-server systems or some standard Graphical User Interface (GUI). Examples of GUI approaches include interfaces such as CMU's Tcl/Tk based interface, HP OpenView, and the World Wide Web (WWW) browser. The WWW browser is currently a popular tool for the NM purpose. §5.5 presents a detailed discussion on the use of WWW browser in NM.

5.3 Interface Modules

In a heterogeneous environment, the NMS has to deal with equipment from different vendors. For example, most ATM gateway switches for enterprise networks in the Ottawa area are the Newbridge 36510, which does not support the SNMP. FORE ATM switches available in the laboratories support SNMP. An intermediary is needed to translate the data between EL or EML and the network managers.

The interface modules are designed to play this role. It translates the management data from a protocol-dependent format in the lower layers into a format defined in the data model, and sends the data to the upper layer managers. The interface modules can be broken down to separate processes as needed in consideration of performance, reliability, and ease-to-implementation.

In design of the interface modules, different approaches have to be taken into consideration because of the software availability and configuration of the software components on different devices.

There are two types of modules (called processes) as shown in Fig.5.2. The event module listens for alarm messages from the EML and EL. An alarm (or trap) is converted to a protocol-independent format and forwarded as an alarm event to the fault manager. The request module performs polling functions. Typically a request is made by the configuration

Figure 5.2 Architecture of a typical interface module.

manager. On receiving the request, the module gets management data from the agent in a protocol-specific format and translates the information to the protocol-independent format and returns the data to the configuration manager.

In the current ATM network setup in NRC, the FORE ATM switch supports a SNMP agent. An SNMP agent should be implemented for a new optical device such as optical switch and amplifier. Because the ATM gateway switch Newbridge's Main Street 36510 has their proprietary database interfaces, either a SNMP agent on top of the switch or an adaptation interface module should be built for the switch to be managed in the new NMS. For devices with SNMP support, the interface modules can be built based on the UCD SNMP's utility *snmptest*, which provides a complete set of SNMP operations. Otherwise the interface modules should rely on the proprietary protocols.

In Figure 5.2, the protocol-independent information format really means a data model in the NMS. The NMS performs various functions on the network data in this model. If the NMS is a proprietary system, the data model can be anything at the designer's willing.

However, if the NMS is to conform to TMN standard, the interface modules have to follow the TMN definitions, as discussed in §2.1.2. For example, if a NE does not comply to TMN, a QA has to be implemented on behalf of the NE. Here, QA is an interface module as required by TMN. Below the QA, a specific protocol is used to talk to the vendor-specific NE. Above the QA, the management information should conform to TMN interface Q3 in order to communicate with the upper layers.

5.4 Network managers

This section discusses three network managers: configuration manager, fault manager, and performance manager. Since the main subject of the thesis is the architecture of the NMS for optical ATM networks, other two well defined areas, i.e., accounting management and security management, are beyond the scope of the project and not discussed here.

5.4.1 Configuration manager

A. Data model

A data model is actually the definition or schema of the database, which defines all relevant entities of the network. These entities represent data components for all processes in the NM layer. Although a formal data model was not yet available during the course of the project, it is necessary to take into account this factor in the design of the NMS. For a complex system, a natural choice of a data model would be object-oriented. A good example of how object-oriented technology is used in implementing a NMS can be found in the literature.^[8]

For the purpose of failure recovery, the configuration manager has to maintain two copies of the NM data. The first copy is stored in virtual memory for fast access. At the meantime, a persistent representation of the same information is stored on disks with minimal overhead. Any changes to the NM data should be automatically written directly to the disk blocks within a database file. A hash table provides efficient access to any object in the database on the disk. When the configuration process is incidentally terminated and restarted late on, the data on the disk is used to rapidly recover the complete state of the network. For a network with tens of ATM switches, a recovery time in an order of 10s should be achieved.^[8]

B. Network discovery

In large optical ATM networks, the components or elements keep changing too quickly to be controlled by the network administrator. A network discovery algorithm is necessarily implemented within the configuration manager modules. Network discovery helps resolve the inconsistency of the network inventory and aids in achieving scalability of the configuration management building blocks.^[8]

The configuration manager discovers inventory directly from the network elements or from element managers where applicable. Now consider the element managers for SONET subnetworks as an example. The network discovery is to collect information on SONET circuit packs, subnetwork connections for SONET double rings, and crossconnections in the endpoint ADMs of point-to-point systems in polling EML systems.

When developing the network discovery algorithm, several issues must be addressed, including access data, location discovery, and status identification, as discussed below.

Access link data, i.e., the connection and traffic information for any pair of two endpoints, still need to be entered into the database manually. There is no known way to automatically discover this information simply because it is difficult to trace the ATM and SONET path. Since a unique ID of each terminating circuit pack can be entered into the SONET protocol header, the direction of the research is to provide reading and writing commands or interfaces at the endpoint.

While it is possible to auto-discover new ATM network elements by polling the SNMP agents, the physical locations of the elements cannot be automatically determined due to the fact that the location entry in the device's SNMP agent is entered arbitrarily. It is hard to know

where the network elements are located just by checking the retrieved data, as shown in the following examples. To demonstrate, consider the results of a SNMP query of the location entry in the MIB-2 of the SNMP agents on four randomly chosen UNIX workstations. Two machines *eureka* and *artie* are in the Network Management Laboratory of Carleton University, Room ME4447 and the other two *alpha* and *omega* are in the Photonics laboratory of NRC, Room 316 of Building M-50. The following are the variable path and the OID for the location entry in the MIB-2 of the SNMP agent:

```
.iso.org.dod.internet.mgmt.mib-2.system.sysLocation.0  
.1.3.6.1.2.1.1.6.0
```

To get the location information, the following query was used

```
snmpget -v 1 host public .1.3.6.1.2.1.1.6.0
```

where *snmpget* is an interface provided by UCD SNMP package and *host* is one of the four machines.

The result for *eureka* was

```
system.sysLocation.0 = "ME4242".
```

It is obvious that this machine was located in "ME4242" when the SNMP agent was setup and the location entry has not been corrected after it was moved to the current location.

artie gives the following response

```
system.sysLocation.0 = "System administrators office"
```

where the information does not tell the correct location information again.

The query for *alpha* results in

```
system.sysLocation.0 = "unknown.".
```

The location entry does not tell the user anything about the location at all. Machine *omega* gives the same result for the location entry.

A human network administrator can probably remember tens or hundreds of locations of network elements in a small network. However, he/she can not be expected to remember the thousands of locations typical in a large network system. When the NMS sends a notice about a particular device, it is much more useful if the information contains the exact location for that device. I believe that it is necessary to promote the standardization of the location entries in the SNMP agent.

The last of the three issues to be solved is the working status of an element. The NMS uses the network inventory data intensively to perform such functions as discovery of the network configuration and fault analysis. The NMS should know the exact working status of the absent network objects, that is, whether the network objects have been intentionally removed, normally disconnected, or some aspects of communication has failed. This problem should partially be solved with the configuration manager and partially solved with the fault manager.

C. Connection management

Making connections is an important function of a configuration manager. For high-speed optical ATM networks, a unique feature of the configuration manager is the automatic routing of the ATM trunks over the SONET network and the routing of a logical fiber link, e.g., a wavelength, over the optical crossconnect switches. Another function is to set up the path for the traffic multiplexing and demultiplexing.

Wavelength routing is done through the crossconnections on the transport nodes in the optical network. In normal operation, the wavelengths are routed to meet the transport needs for the whole optical network. At the same time, the crossconnect switches can also provide flexibility to meet special needs. For example, a world fair or Olympic games needs a very high bandwidth during a specific period of time. In this case, the wavelengths are reconfigured and higher bandwidth is reserved. In addition, the crossconnect also provides easy access to reroute the optical path for network extensions or failure protection purposes.

Traffic multiplexing/demultiplexing can be done for a given wavelength configuration by ATM switching and the ADM mechanism. ATM switches that are connected to the optical access node can be used for user data multiplexing and demultiplexing due to the ATM switching flexibility. Many end user devices can be directly connected to the ATM switch at each optical node.

Switching in optical networks is a different concept than that in ATM networks and it deserves more clarification. In ATM networks, switching technology falls into two categories: PVC and SVC. At present, most switches support PVC only. The user has to make the connection manually before communications can take place. Although SVC is the ultimate target for the ATM switching technology, only some ATM switches currently support SVC. The ATM NM still must provide connection management service to the users. The procedure includes routing and bandwidth allocations; both should be done on the ATM switches.

In the case of an optical network, a user accesses the network service through ATM switches or some dedicated access node such as an optical terminal.^[6] The terminal has to be connected to the all-optical network, usually via an ADM node. At the destination, the traffic

has to be demultiplexed by changing the tunable electronic wavelength selector. If the optical path is not ready by the time the connection request is made, the connection manager has to route the path from the user end to the destination by configuring the wavelengths through multihops. A generic procedure for making a connection in optical networks consists of several steps^[8]

1. Automatically choose the appropriate access node at each end.
2. Compute an end-to-end route over the optical network. An effective routing algorithm is in demand.
3. Correlate the circuit pack information from the connection request to the network switching elements.
4. Select unused high-speed channels.
5. Send the resulting crossconnect information to the network elements or SONET subnetwork element managers.
6. Correlate all received confirmations on the connection requests.
7. If successful, put the connection into service and inform the user who initialized the request.

5.4.2 Fault manager

Fault diagnosis is the major function a fault manager performs. First, the fault manager receives network alarms in the defined data model. It then carries out the fault analysis on the managed objects. Finally it presents the results to the user via an appropriate GUI.

An important feature of the fault management is the integrated root-cause analysis over the ATM, SONET, and fiber connections. This analysis takes advantage of the configuration management modeling capability that correlates different network technology layers. Alarms from multiple nodes are correlated, analyzed, and displayed to the user. In a complex system, this display can be coded in color for a better vision.

To facilitate alarm correlation, the fault manager dynamically builds and maintains a managed object model with queries to the configuration manager. The managed objects are represented in the fault manager as objects in a containment tree. As mentioned early, an event in a management-protocol-dependent format is first converted to the internal format at the interface modules. It is then sent over to the fault manager in a consistent format. Finally a relevant algorithm become active to perform the diagnosis.

5.4.3 Performance manager

In addition to traditional functions of performance management and monitoring the physical parameters of the optical elements, one important function of the performance manager is the test function. For optical networks, all accesses for the user should specially designed. For example, loop back is often used to test the communication path. On a crossconnect, loop back is realized by the pass-through configuration. To test an optical path, the end user first finds the optical path and nodes on the path. The path is tested node by node. On each crossconnect switch being tested, the connection is made on the pass-through state.

5.5 WWW and GUI

5.5.1 Fundamentals

As part of the NMS, the GUI is a tool that allows the user to interact in a clear and elegant way with the network manager software such as configuration manager and fault manager. A GUI should display network maps, device faceplates, alarms, connections, and other visible network entities. Multiple instances of the GUI can be operated on separate workstations where each user may control some aspects of the network view independently. A dedicated process should control the user's accesses, coordinate the different views, and cache the network data to reduce the communications burden on other modules and interfaces.

A GUI can be implemented using various languages and environments. For example, the GUI in CMU/UCD SNMP package is written in Tcl/Tk.^[31] HP OpenView and many other "platforms" may also be used as a GUI. Everything in the Advent SNMP package is written in Java. However, what is interesting for the author is the approach of the Web-based NMS.

Web-based management is the application of World Wide Web (WWW) tools for NM. It uses HTTP (Hypertext Transfer Protocol) servers for providing static, dynamic, and interactive contents of management information, and uses the Web browser as the GUI. Appendix A.3 presents a description of the HTTP servers in detail. An HTTP server acting in a management role can provide information in a variety of forms, including HTML (Hyper Text Markup Language), graphics, binary encoded information, and executable code that are programs typically written in C/C++, Perl, Visual Basic or JAVA. At the same time, widely available Web browsers are a natural choice for a network-wide GUI. In short, Web technology allows

HTTP to function as a possible protocol between the GUI and the network managers and allows Web browsers to function as a GUI of the NMS.

Using HTTP to export management data from network manager processes to the WWW browser results in many advantages over a traditional NMS. First, no specialized GUI is needed to display the management data. At the client (user) side one can simply use a Web browser. On the management service side, all that is needed are the light weight interfaces on the HTTP server. Second, the versioning problems are eliminated. These problems would typically occur when an old agent or manager does not support the new and possibly required features of the other. By using a Web-based approach, the service and browser side do not need to be updated simultaneously. Another major advantage is platform and location independence for management services. A network administrator can access the NM services from anywhere, on any platform. All that is needed is a general-purpose Web browser, which is standard software and freely available for the vast majority of hardware/operating systems. A final major advantage is its seamless integration capability with on-line documentation. Context-sensitive help and documentation may be accessed via hyperlinks embedded directly into the management pages. Additionally, configuration and management can be driven entirely from an on-line instruction manual.

There are also some drawbacks or concerns in applying the Web-based approach. The first issue is the latency of HTTP for small transactions. HTTP is layered over TCP and the user messages are usually transported to the HTTP server first and then passed to the application program. For short messages, the overhead is relatively large. Another issue is that the Web-based approach places more burden on the software of the network managers.

The above rationale shows that the usage of Web-based NMS offers much potential and its implementation is more cost-effective than a traditional management system. The next subsections describes the software components in a Web-based management system.

5.5.2 Software components

The basic software components for a WWW GUI are a WWW browser and a HTTP server. Today almost every computer has a WWW browser as built-in software.

In addition to the WWW browsers and servers themselves, several firmware packages have been developed recently. These packages fall into two categories for the needs of Web-based NM. The first category is middleware between the Web browser and the SNMP agent. Examples include DR-Web, WebSNMP and Webbin'CMIP. These packages include a HTTP server and an interface between the server and device agent. Using this category, the network administrator has more control over the device access.

The other category is a MIB browser that can be used to browse the variables of the MIB. Examples include SNMP MIB Browser, MibMaster, NetDirector@Web and NetBrowser. The SNMP MIB Browser was tested on a known host and MibMaster was tested on the built-in host. Both tests were successful. All of these packages are Web/SNMP agent based. It is obvious that our need is one step further than these firmware. An interface between the HTTP server and the NM services is especially needed. This issue is addressed in the following paragraphs in this section.

When the HTTP is used to control and monitor the network manager processes, the server has to interact with the services via a CGI file. A CGI file is an executable program that

runs on demand. Typically a CGI file is written in C/C++, Perl, Visual Basic, or other languages. The NM data for display can be obtained by running the CGI program and the results are packed up and sent back to the browser via the HTTP server or by the CGI program directly. Usually the interactions between the HTTP server and the browser is done using HTTP forms.

The approach described above is a CGI/HTML solution. It is relatively simple and cost effective in the short term. However it is hard to use a series of form-based transactions to implement a complex and interactive display system. To achieve such goal, Java has the capability to export the live GUI through the Web as a dynamically acquired component. Java is not simply a replacement of other languages such as C/C++ and Perl for programming the same architecture of CGI; instead, it has a unique characteristic of platform neutrality. Although there exist many different approaches in using Java for Web applications, a widely-accepted approach is to invoke Java applets within the Web browser on demand and run serving applets at the server. One can choose to run either a HTTP service or Java-related service. This depends only on which Web pages is selected.

By developing various service interfaces at the server side, the Java applets at the server can interact with the network manager processes. These applets can run on any kind of computers as long as they have a Java virtual machine environment. On the browser side, the only requirement is that the browser be Java-enabled.

It should be pointed out that the Java approach is not specific to any particular protocol. The communications between the services on the HTTP server and the network manager processes can rely on any of the protocols such as DMI and RMI (Remote Method

Invocation). The connections between the Web browser and the server can be established using any inter-object communications interfaces such as CORBA. The unique benefit of using Java is that one can run the same piece of software on different machines. This is the basis of Java's reputation for software reuse and applicability in a distributed and heterogeneous environment.

Several Java/Web-based NM packages have been released, such as Advent Java SNMP,^[33] JUMP,^[34] and JMAPI.^[35] They are a good start pointing for the proposed NMS architecture.

In summary, an architecture for optical ATM NMS was proposed in this chapter. A TMN-based functional hierarchy was adopted for the architecture in order for the system to work in a heterogeneous environment. The management functions and the interface modules were described. A Web-based GUI was suggested and various issues were discussed in detail.

Chapter 6

CONCLUSIONS AND FUTURE WORK

This thesis presents the results of my research on the management of high-speed optical ATM networks with an emphasis on the NMS architecture. It is organized in three parts: experiments on management tools, exploration of the high-speed optical ATM networking technologies, and a proposal of an NMS architecture.

6.1 Experimental Results

Experimental studies have been conducted on a SNMP agent development kit, SNMP manager software package, and client/server systems written in different network APIs.

IBM's SystemView DPI/DMI Agent kit has shown to be a suitable tool for developing SNMP agents and subagents. Its extensibility and ease of implementation has been highly appreciated. On the one hand, a DMI subagent serves as an interface between the DPI-enabled

SNMP master agent and the DMI components. Managed resources are instrumented once and may be managed by any DMI applications. On the other hand, multiple DPI subagents can be registered to the master agent locally or remotely. In the experiment, a stand-alone DPI subagent (other than the DMI subagent) has been successfully hooked up to the DPI master agent. This agent/subagent architecture is especially useful for distribution of the NMS functionality.

During the course of the project, a DMI subagent was developed for the optical crossconnect switch, which has a proxy agent structure. Within the agent, port connections, current, temperature, broadcast and bypass connection methods were instrumented. The subagent was implemented as a *Direct-interface* program so that the switch process runs continuously, providing a faster response than an equivalent *Overlay* implementation. SNMP functions *get*, *getnext*, and *set* were implemented and tested successfully.

The UCD/CMU SNMP manager is a highly portable package and was successfully installed and tested during the project. The SNMP functions *snmpwalk*, *snmpget*, *snmpgetnext*, and *snmpset* were found to be working properly. The module *snmpctest* is an API for the user to make queries for high level applications. The SNMP package has proven to be a good starting point for developing SNMP-based management applications.

Client/server systems were written in BSD sockets and with FORE Systems' ATM-API. The performance was analyzed based on three metrics: round trip latency, throughput, and jitters. The latency for TCP sockets on Ethernet strongly depends on the network configuration. While the latency for the two Sparcstations on the NRC net behaved normally, the latency for the SGI machines has peaks of a regular shape. This was explained in terms of

the network MTU and the TCP sliding window. The socket option *TCP_NODELAY* can be used to control the occurrence of the peaks by invoking the function *setsockopt()*. Another way to turn around this problem is to manually pad the unfull segment to a full segment in the application. Similar results were obtained for TCP/IP/ATM. The latency of the Ethernet rapidly increases at messages around 15 kbytes. The latencies for TCP/IP/ATM and the ATM-API are close to each other. The latter has much smaller fluctuations and thus a much more predictable performance than the former.

As for the throughput, a message around 3 kbytes can achieve the expected value (20 Mbps) for ATM-API. For TCP/IP/ATM, only messages larger than 17 kbytes can achieve the expected data rate. At messages around 10 kbytes, a large inverse-triangle-shaped dip exists, which is undesirable for real-time applications. For TCP/IP/Ethernet the throughput reaches its maximum of 7 Mbps at around 12 Kbytes and then drops to about 3 Mbps. This is mainly due to the MAC contention.

The ATM-API does not have a sliding window for flow control and thus the user does not have to worry about the problem of the 200ms time-out as is the case of using sockets. Due to the special design of the ATM format (small cells) and switching strategy, the fluctuations are much smaller than that in the case of sockets. Thus the ATM-API makes the communication latency more predictable. However, the user has to manage the application level segmentation and flow control manually. The SSCOP protocol is strongly suggested in order to have a high-speed and yet a reliable ATM network.

6.2 High-Speed ATM Optical Networks

The target network in this research is based on the double ring structure for local connections and all-optical transport networks for backbone infrastructure. The optical fiber rings interconnect the ATM switches through dedicated access nodes. The access nodes route the ATM traffic based on the information in the ATM header on a cell-by-cell basis, independently of the ATM switches. All ATM cells travel along the same fiber path and in the same wavelength. ADM together with ATM switch is used as user access to the optical networks. The ATM switches are only loaded by the traffic that is originating from or destined to a site covered by the switch.

In multi-hop optical transport networks, WDM technology is used to shift the routing workload from ATM switches to the optical transport nodes. ATM cells make multiple hops through intermediate optical nodes to reach their destinations. This reduces the load on the switches because data traffic not destined for a given node does not have to pass through its switch's electronics; instead it may be routed optically to the next node. The network resilience is improved through the ability to offer alternative data traffic routes. The fiber between two intermediate nodes may carry traffic in multiple wavelengths. As an ATM cell travels from the source to the destination, it may undergo different wavelengths.

In the optical transport networks, each node incorporates an optical cross-connect switch, so that the overall interconnection of the nodes can be reconfigured to support particular network requirements. A major feature of such networks is the reuse of the wavelength. Within each node, a combination of wavelength-selective elements together with

an optical space switching can assign an input wavelength to any outgoing fiber or drop it to the local ADM node.

6.3 Next Generation NMS

As discussed in Chapter 5, a TMN-like functional hierarchy is a natural choice for a NMS in order to manage diverse network elements for large-scale optical ATM networks. The key issues discussed were the data model, multitiered and multiprotocol architecture, integration capabilities, network discovery, fault diagnosis, and the Web-based GUI approach.

The data model is a direct output of the network configuration management. Based on this data model, other NM applications such as fault diagnosis and performance monitoring can be implemented. A major concern of a management system for large networks is the scalability and performance due to the centralization of the databases. With the multiple layer approach, it is possible to distribute the database across multiple workstations and subnetwork element managers. Care must be taken for database consistency and synthesis in network-wide applications such as integrated fault root-cause analysis and end-to-end connection management.

Making connections is particularly important to high-speed optical ATM networks. A unique feature of the configuration manager is the automatic routing of ATM trunks in multitechnology networks. Wavelength routing is done through the crossconnects on the transport nodes in the optical network. The wavelengths are routed to meet the transport needs for the whole optical network. In the meantime, the crossconnect switches can also provide flexibility to meet special needs.

Interface modules are designed to mediate the differences of multiple protocols. They translate management data between protocol-dependent formats and an internally defined format. In the SNMP domain, the interface modules can be built on top of UCD SNMP API *snmptest* that provides a complete set of SNMP operations. In addition, CORBA is an integration environment, which can be taken considered in the future. CORBA can be used as “glue” in the application layer where any applications are defined in IDL interfaces. In this approach, the interfaces and implementations are well separated, making the integration much easier than a traditional NMS.

For network inventory recovery, a configuration manager has to maintain a copy of the NM data in virtual memory and a second copy on the disks. A network discovery algorithm is necessarily implemented within a configuration manager module to help solve the inconsistency of the network inventory. In addition, it is important to promote the standardization of the system location entries in the SNMP agent. Also the working status of an element should be solved properly to let the network operator to know if the element is intentionally removed, normally disconnected, or something failed.

Fault diagnosis is the major function of the fault manager. First, the fault manager receives network alarms. Then it carries out the fault analysis on the managed objects. Finally it presents the results to the user via a GUI. An important feature of the fault manager is the integrated root-cause analysis over the ATM, SONET, wavelengths distribution, and fiber connections. Configuration management modeling capability can be used to correlate different network technology layers. Alarms from multiple nodes are correlated, analyzed, and displayed to the user.

The Web-based management approach uses the Web browser as the GUI and uses HTTP servers for providing static, dynamic, and interactive contents of management information. The server interacts with the services via CGI, which is typically written in languages such as C/C++, Perl, Java, and Visual Basic. The NM data for display can be obtained by running the CGI program on the device, which packs the results and sends them back to the browser. The advantages of using Web-based NM approach include elimination of versioning problems, platform and location independent, and seamless integration capability. Utilization of Web-based NMS offers much potential of productivity and its implementation is more cost-effective than a traditional NMS. The connections between the Web browser and the server can be established using any inter-object communications interfaces such as RMI and CORBA. In the meantime, Java's applets provide more freedom to both client side and server side. The unique benefit by using Java is that one can run the same piece of software on different machines. Java's software reuse and applicability adds more value to the NMS in a distributed and heterogeneous environment.

APPENDICES

A.1 Optical Switch MIF

The MIF of the optical switch was defined as following:

Start Component

Name = "Optical Switch - Direct Interface"

Description = "This MIF is for optical switch controlled by NT via RS232."

Start Paths

Name = "RS232DIR"

WIN32 = "D:\\SVA\\DM\\EXAMPLES\\RS232DIR\\RS232DIR.EXE"

os2 = direct-interface

End Paths

//Enumerations for ComponentID group

Start Enum

Name = "Verify_Type"

Type = Integer

0x00 = "An error occurred; check status code"

0x01 = "This component does not exist"

0x02 = "The verify is not supported"

0x03 = "Reserved"

0x04 = "This component exists, but the functionality is untested"

0x05 = "This component exists, but the functionality is unknown"

0x06 = "This component exists, and is not functioning correctly"

0x07 = "This component exists, and is functioning correctly"

End Enum

Start group

name = "ComponentID"

id = 1

class = "DMTF|ComponentID|1.0"

description = "This group defines attributes common to all components. This group is required."

Start attribute

name = "Manufacturer"

id = 1

description = "The name of the manufacturer that produces the component."

access = READ-ONLY

storage = COMMON

type = STRING(64)

value = "Carleton and NRC."

End attribute

Start attribute

name = "Product"

id = 2

```
description = "The name of the component."  
access = READ-ONLY  
storage = COMMON  
type = STRING(64)  
value = "Optical switch"  
End attribute
```

Start attribute

```
name = "Version"  
id = 3  
description = "The version for the component."  
access = READ-ONLY  
storage = COMMON  
type = STRING(64)  
value = "Prototyping version 1.0"  
End attribute
```

Start attribute

```
name = "Serial Number"  
id = 4  
description = "The serial number for this instance of this component."  
access = READ-ONLY  
storage = SPECIFIC  
type = STRING(64)  
value = "96-09-20"  
End attribute
```

Start attribute

```
name = "Installation"
```

```
id      = 5
description = "The time and date of the last install of this component."
access  = READ-ONLY
storage = SPECIFIC
type    = DATE
value   = ""
End attribute
```

Start Attribute

```
Name = "Verify"
Id = 6
Access = Read-Only
Storage = Specific
Type = "Verify_Type"
Description = "A code that provides a level of verification that the component is still
installed and working."
Value = 0x07
End Attribute
```

Start attribute

```
name    = "Workstation_name"
id      = 7
description = "The name of the work station that communicates with the switch."
access  = READ-ONLY
storage = COMMON
type    = STRING(64)
value   = "rho.ps.iit.nrc.ca(NT3.51)."
End attribute
```

Start attribute

```
name  = "SerialPort"
id    = 8
description = "The serial port for the communication channel between workstation and
mcricontrollor."
```

```
access = READ-ONLY
```

```
storage = COMMON
```

```
type   = STRING(64)
```

```
value  = "COM1"
```

End attribute

Start attribute

```
name  = "CommunChannel"
```

```
id    = 9
```

```
description = "The name of the communication channel."
```

```
access = READ-ONLY
```

```
storage = COMMON
```

```
type   = STRING(64)
```

```
value  = "9-pin-RS232"
```

End attribute

Start attribute

```
name  = "Microcontrollor"
```

```
id    = 10
```

```
description = "The mode of Microcontrollor."
```

```
access = READ-ONLY
```

```
storage = COMMON
```

```
type   = STRING(64)
```

```
value  = "MC68HC11E9"
```


End attribute

Start Attribute

    Name = "NumOfPorts"

    ID = 11

    description = "Number of ports of this switch."

    access = READ-ONLY

    storage = COMMON

    Type = INTEGER

    value = 16

End Attribute

Start attribute

    name = "People"

    id = 12

    description = "The people who used IBM's DPL/DMI and wrote this software."

    access = READ-ONLY

    storage = COMMON

    type = STRING(64)

    value = "Guowei Zhang"

End attribute

End group

Start Group

    Name = "Connects"

    ID = 2

    Class = "DMTFConnects1.0"

Description = "Optical switch with direct interface instrumentation"

Key=1

Start Attribute

Name = "Index"

ID = 1

Description = "Index into port table."

Access = READ-ONLY

Type = INTEGER

Value = *"RS232DIR"

End Attribute

Start Attribute

Name = "PortStatus"

ID = 2

Description = "Operation status of this port (down/ operational)."

Access = READ-WRITE

Type = DISPLAYSTRING(32)

Value = *"RS232DIR"

End Attribute

Start Attribute

Name = "Power"

ID = 3

Description = "Power of this port."

Access = READ-ONLY

Type = INTEGER

Value = *"RS232DIR"

End Attribute

Start Attribute

Name = "Temperature"

ID = 4

Description = "Temperature of this port."

Access = READ-ONLY

Type = INTEGER

Value = *"RS232DIR"

End Attribute

Start Attribute

Name = "OutputPortForCurrent"

ID = 5

Description = "To set an output port number as a flag in order to measure current between this input port and the output port specified here."

Access = READ-WRITE

Type = INTEGER

Value = *"RS232DIR"

End Attribute

Start Attribute

Name = "Current"

ID = 6

Description = "Current between this input port and the output port set by OutputPortForCurrent."

Access = READ-ONLY

Type = INTEGER

Value = *"RS232DIR"

End Attribute

Start Attribute

Name = "BroadcastStatus."

ID = 7

Description = "The input port that broadcasts (on/off)."

Access = READ-WRITE

Type = DISPLAYSTRING(32)

Value = *"RS232DIR"

End Attribute

Start Attribute

Name = "PassThroughStatus."

ID = 8

Description = "Pass through status: (on/off)."

Access = READ-WRITE

Type = DISPLAYSTRING(32)

Value = *"RS232DIR"

End Attribute

Start Attribute

Name = "ConnectTo."

ID = 9

Description = "Which port this port is connected to."

Access = READ-WRITE

Type = INTEGER

Value = *"RS232DIR"

End Attribute

End Group

End Component

A.2 SystemView Agent Kit Set Up

For Windows NT/95 system, three files should be downloaded: *README*, *LOADDSKF.EXE*, and *SystemView Agent for Windows NT/95* (file name: *SVAPAC.ZIP*)

Unpacking the compressed file from DOS prompt

PKUNZIP SVAPACK.ZIP (or use WinZIP)

gives three disk image files:

SVADISK1.DSK, n=1, 2, 3.

Then the three disk image files should be expanded and saved to three floppy disks with the *LOADDSF* executable, say, at driver a:

LOADDSKF SVADISKn.DSK a: /F

Where $n=1,2,3$.

Setup procedure starts by inserting the first disk and typing "A:\setup", then follow the instructions there. After a successful installation, two PostScript documents as manuals can be found from the root installation directory:

SystemView Agent for Win32 User's Guide (SVAUSERW.PS)

SystemView Agent DMI Programmer's Guide (SVADMIPG.PS)

If the SystemView Agent is to be used with a remote SNMP application, UDP transport-layer support is required. This support is provided through the Windows NT/95 operating system. To make the SystemView Agent ready to listen for the requests from a remote SNMP application, three executables are needed:

SNMP master agent (SNMPD)

DMI subagent (DMISA)

Extensible agent (EXTAGENT)

The ***SVASTART.BAT*** file is provided with the SystemView Agent for Win32 as a convenient way to execute the three executables on Windows NT/95 workstation. This batch file can be added to the Startup group or executed manually.

In addition to the DMI package, IBM also provides a DPI 2.0 API freeware package. It contains a developers toolkit for DPI subagent developers and or programmers who want to include the DPI 2.0 protocol in their SNMP agent. This toolkit includes DPI 2.0 RFC (RFC1592) , programmer Reference, and subagent sample code. The software is located at FTP site:

software.watson.ibm.com/pub/dpi

A.3 HTTP

HTTP was designed to be a protocol with the lightness and speed which are necessary for a distributed hypermedia information system. It is a generic stateless object-oriented protocol. When used on the Internet, the HTTP communications take place over a TCP/IP connection in a client/server model. Here the client is the WWW browser and the server means the HTTP server. A basic HTTP transaction consists of four steps

1. The client establishes a connection to the server at a well known port number with 80 as the default;
2. The client sends a request message to the server;

3. The server gets the data accordingly and sends a response to the client at a non-reserved port number specified by the client at the time of request;
4. The client closes the connection.

A key feature of HTTP is the negotiation of data representation, allowing systems to be built independently of the development of new advanced representations. This is accomplished using the MIME (Multipurpose Internet Mail Extensions).^[Bor]

The following figure shows how the HTTP client/server model is implemented for transactions between the client and the server. The client accesses the server by means of a new common addressing scheme known as URL (Uniform Resource Locators). The syntax of a general URL has a form of

Figure 5.3 HTTP client/server model and CGI-based management architecture.

scheme://host:port/pathname

where *scheme* can be any of *http*, *gopher*, *telnet*, *ftp*, and other services, *host* is the host where the HTTP server resides, *port* is the port for the HTTP server, and the *pathname* is the name of the file that the client is invoking.

HTTP servers are dominated by two major players: NCSA (National Center for Supercomputing Applications) and CERN (The European Center for High Energy Physics). The information of the resources and installations of the two HTTP servers are described in detail in the literature.^[32] Below is a brief discussion of the main features the HTTP servers possess. That will be helpful in building up a WWW GUI for an optical NMS.

NCSA *httpd* 1.5 is the current version and it supports many attractive features. The server can map virtual URL paths to physical paths on the server and determines the MIME type of the served document and send it to the client. The server can generate directory listings, control access to directories, and serve users' web page. It serves different documents even on the same physical machine (IP address binding). The server parses HTML documents and executes the relevant scripts. The server can run multiple processes to serve documents required by different clients simultaneously for faster response.

Another major HTTP server was implemented by CERN and the current version is 3.0. There are several differences between the CERN and NCSA server. The CERN server can function as a cached proxy, has better path mapping to allow for shorter URLs, has no server-side includes, and can handle content negotiations between the client and the server. On the other hand, the NCSA server has better and more easily configurable access control.

REFERENCES

1. F. Masetti, J. Benoit, F. Brillouet, et al, "High Speed, High Capacity ATM Optical Switches for Future Telecommunication Transport Networks", *IEEE J. Selected areas in Comm.*, Vol. 14 No. 5 (1996) 979.
2. Ernst Munter, James Parker, and Paul Kirkby, "A High-Capacity ATM Switch Based on Advanced Electronic and Optical Technologies", *IEEE Comm. Mag.*, Nov. (1995) 64.
3. Th. M. Martinson, "Corporate ATM Optical Transport Network", *5th Annual Conf. European Fiber Optic Comm. and Networks* (The Hague, June 30 - July 2, 1993) 166.
4. I.P. Kaminow, C.R. Doerr, C. Dragone, et al, "A Wideband All-Optical WDM Network", *IEEE J. Selected areas in Comm.*, Vol. 14 No. 5 (1996) 780.
5. Douglas Marquis, Steven G. Finn, and Salil A. Parikh, "Network management for all-optical networks", *Optical Fiber Communication, Technical Digest Series*, (Optical Society of America, Washington, D.C., 1995) 166.
6. D. Marquir, S.A. Parikh, R.A. Barry, et al, "Control and Interfaces for an All-Optical Network Testbed", *IEEE Interl. Conf. Comm.*, (1996) 1586.
7. S. Bajaj, N. Cheung, G. Hayward, and Y. Tsai, "High-Speed ATM/SONET Infrastructure Research in ATDNet", *IEEE Network*, July/August (1996) 18.
8. R. Doverspike, M. Maeda, S. Narain, et al, "Network Management Research in ATDNet", *IEEE Network*, July/August (1996) 30.

9. William Stallings, *SNMP, SNMPv2, and CMIP, The Practical Guide to Network-Management Standards* (Addison-Wesley, New York, 1993).
10. Roch H. Glitho and Stephen Hayes, "Telecommunications Management Network: Version vs. Reality", *IEEE Comm. Mag.* (1995) 47.
11. CCITT Recommendation M.3010, "Principles for a Telecommunications management Network", Geneva, 1992.
12. RFC 1592, SNMP Distributed Protocol Interface (DPI), Version 2.0.
13. IBM Corporation, *SystemView Agent for Win32 User's Guide* (document in the SystemView Agent kit, available on Web site: <http://www.software.ibm.com/sysman/download/casdk.html>).
14. Banyan Systems Inc., Network Management White Page, <http://www.banyan.com/netmgt/netmgt.html>.
15. <http://www.cis.ufl.edu/~dadavis/cmu-snmp2-src.uuencode>.
16. <ftp://ftp.ece.ucdavis.edu:/pub/snmp/ucd-snmp.tar.gz>.
17. T. Luckenbach, "Performance experiments within local ATM networks", 6th Annual Conf. *European Fiber Optic Comm. and Networks* (Geneva, 1994) 166.
18. T.R. Henderson, "Design Principles and Performance Analysis of SSCOP, A New ATM Adaptation Layer Protocol," *Computer Cmmun. Review*, Vol. 25 (April 1995) 47.
19. ITU-T Q.2110, "Service Specific Connection Oriented Protocol".
20. P.E., Green, "Optical Networking Update", *IEEE J. Selected areas in Comm.*, Vol. 14 No. 5 (1996) 764.
21. Jean-Yves Le Boudec, "The Asynchronous Transfer Mode: a tutorial", *Computer Networks and ISDN Systems*, 24 (1992) 279.
22. Andrew S. Tanenbaum, *Computer Networks* (Prentice Hall, New Jersey, 1996).
23. Martin De Prycker, *Asynchronous Transfer Mode, Solution for Brodband ISDN* (Prentice Hall, New York, 1995).
24. J. McEachen, "Gigabit Networking on the Public Transmission Network", *IEEE Comm. Magazine*, Vol. 30, No. 4 (1992).

25. M. J. O'Mathony, "Optical Multiplexing in Fiber Networks: Progress in WDM and OTDW", *IEEE Comm. Magazine*, Dec. (1995) 82.
26. B. Petri, "Signaling support for multimedia communications in ATM networks", *6th Annual Conf. European Fiber Optic Comm. and Networks* (Neneva, 1994) 18.
27. P.E. Green, *Fiber Optic Network*, Englewood Cliffs, NJ: Prentice Hall, 1993.
28. C., Partridge, *Gigabit Networking.*, Reading, MA: Addison-Wesley, 1994.
29. Gigabit networks. *special issue of IEEE Comm. Mag.*, Vol. 30, No. 4, 1992.
30. T. Martinson, C.J. Moss, "High-Speed Optical ATM Backbone for Campus Language Learning", *13th Annual Conf. European Fiber Optic Comm. and Networks* (Brighton, England, June 27 - 30, 1995) 69.
31. B.B. Welch, *Practical Programming in Tcl and Tk* (Upper Saddle River, NJ: Prentice Hall, 1995).
32. Yuval Fisher, *Spining the Web* (Springer-Verlag, New York, 1996).
33. <http://www.adventnet.com/tutorial.html>.
34. <http://192.135.184.58/products/jump/>.
35. <http://java.sun.com/products/JavaManagement/>.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved