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

NETWORK CONFIGURATION MANAGEMENT IN HETEROGENEOUS ATM ENVIRONMENTS

Yanrong Li

Thesis Submitted to The Faculty of Graduate Studies and Research
in Partial Fulfilment of the Requirements
for the Degree of Master of Science

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

January 1998
©Yanrong Li

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.

0-612-27019-X

Canada

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis

Network Configuration Management In Heterogeneous ATM Environments

submitted by

Yanrong Li

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

Prof. B. Pagurek
Thesis Supervisor

Dr. Rafik A. Goubran, Chair,
Department of Systems
and Computer Engineering

Carleton University
January 1998

Network Configuration Management In Heterogeneous ATM Environments

Yanrong Li

Abstract

The configuration of Permanent Virtual Connections (PVCs) in a heterogeneous ATM network is difficult for a network operator. Each ATM vendor provides their own way to configure a PVC for its ATM switches. There is no uniform way for the network operator to set up an end-to-end PVC in a heterogeneous ATM network without knowing the vendor-dependent configuration details of each of the switches.

This thesis work covers the design and implementation of a generic model based upon Mobile Agents to perform PVC configuration management functions in multi-vendor ATM networks. Using a simulation testbed written in the JAVA programming language, the thesis introduces an alternative or complement to any existing ATM configuration management solutions. A prototype implementation is developed for the validation of the proposed solution.

Acknowledgements

I would like to take this opportunity to show my gratitude to those helped me, supported me and encouraged me during my thesis work. First all, I would like to thank Professor Pagurek, my supervisor, who provided me great encouragement during my stay, and TRIO for the financial support. I would like to thank Professor Bieszczad, who helped me in getting my thesis off the ground. I also would like to thank Gatot Susilo, for his general help in my implementation. In addition, I would like to thank the Photonic Systems Group and the SPIN Group of IIT in NRC, for providing me their facilities. Above all, I would like to thank Richard Clark, who was there when I needed him the most.

Table of Contents

Chapter 1 Introduction	1
Chapter 2 Background Knowledge	3
Section 2.1 ATM Overview	3
Section 2.1.1 ATM Cell Structure	3
Section 2.1.2 ATM Architecture	5
Section 2.1.3 VP and VC Switching	7
Section 2.1.4 Quality of Service and ATM Service Classes.....	10
Section 2.2 ATM Network Management	13
Section 2.2.1 Network Management Framework	13
Section 2.2.2 Functional Areas of Network Management	15
Section 2.2.3 Network Management Protocols	17
Section 2.2.4 Management Information Base	19
Section 2.3 Mobile Code Based Network Management	20
Section 2.3.1 Motivation for a New Approach to Network Management	21
Section 2.3.2 Framework of Network Management with Mobile Code	23
Section 2.3.2.1 Infrastructure of Mobile Code	23
Section 2.3.2.2 Network Simulator	25
Section 2.3.3 PVC Configuration Application Based on Mobile Code	27
Chapter 3 ATM PVC Configuration Management....	29
Section 3.1 PVC Configuration Requirements	29
Section 3.2 PVC MIBs	30
Section 3.2.1 ATM Forum ILML 4.0 MIB (UNI 3.1)	31
Section 3.2.2 IETF ATM MIB (RFC 1695)	36
Section 3.2.3 Fore Runner Switch MIB	44
Section 3.2.4 Cisco LightStream Switch MIB	50
Section 3.3 Common View of PVC MIB	52

Chapter 4 Different Approaches to PVC Configuration Management	55
Section 4.1 Manager to Manager Based PVC Configuration Management	55
Section 4.2 CORBA Based PVC Configuration Management	59
Section 4.3 Mobile Code Based PVC Configuration Management	62
Chapter 5 PVC Configuration Management with Mobile Code in Heterogeneous ATM Environment	64
Section 5.1 Architecture of Mobile Agent Based PVC Provisioning ...	64
Section 5.2 Application Software Requirements	69
Section 5.3 Application Software Analysis	71
Section 5.3.1 Mapping PVC Common View to Vendor Specific MIB	71
Section 5.3.2 Scenario Analysis	73
Section 5.4 High Level Design	77
Section 5.4.1 Software Components	77
Section 5.4.2 Class Diagrams	79
Section 5.5 Detailed Design	83
Section 5.5.1 Setting Up End-to-end PVC Scenario	83
Section 5.5.2 Tracing End-to-end PVC Scenario	84
Section 5.5.3 Releasing End-to-end PVC Scenario	84
Section 5.6 Implementation	88
Section 5.7 Testing Examples	88
Chapter 6 Conclusions	93
Section 6.1 Summary	93
Section 6.2 Future Work	94
References	96

List of Tables

Table 2.1 ATM QoS parameters	10
Table 3.2.1.1 Physical Port Group	34
Table 3.2.1.2 ATM LayerGroup	34
Table 3.2.1.3 Virtual Path Connection Group	35
Table 3.2.1.4 Virtual Channel Connection Group	36
Table 3.2.2.1 Interface Configuration Parameters Group	38
Table 3.2.2.2 Interface Virtual Path Link Group	38
Table 3.2.2.3 Interface Virtual Connection Link Group	39
Table 3.2.2.4 Virtual Path Cross Connection Group	41
Table 3.2.2.5 Virtual Channel Cross Connection Group	43
Table 3.2.3.1 Port Group	46
Table 3.2.3.2 In Path Group	47
Table 3.2.3.3 Path Route Group	48
Table 3.2.3.4 Out Path Group	49
Table 3.2.3.5 Channel Route Group	49
Table 3.2.4.1 Line Interface Group	51
Table 3.2.4.2 VPC Connection Group	52
Table 3.3 PVC MIBs and Common View	54

List of Figures

Figure 2.1 ATM Cell	4
Figure 2.2 B-ISDN protocol reference model	6
Figure 2.3 ATM Switches: Virtual Path and Virtual Channel Switchces	8
Figure 2.4 Virtual path switching	9
Figure 2.5 Manager/Agent Model	14
Figure 2.6 Object Identifier Tree	20
Figure 2.7 Infrastructure of Mobile Code	24
Figure 2.8 Network Simulator Architecture	26
Figure 3.1 ATM Forum UNI ILMI MIB	33
Figure 3.2 IETF ATM MIB	36
Figure 3.3 Fore ATM MIB	45
Figure 3.4 Cisco LightStreamSwitch MIB	50
Figure 4.1 Conceptual Model of Multi-domain Management	56
Figure 4.2 Example: An ATM Network Management Tool	57
Figure 4.3 An Architecture for Management System in Heterogeneous ATM networks.	61
Figure 4.4 PVC Configuration Management Based upon Mobile Code	62
Figure 5.1 ATM PVC Path	65
Figure 5.2 Non Mobile Code vs. Mobile Code Way of Configuring a PVC	67
Figure 5.3 A Simulated ATM Network	68
Figure 5.4 Model of Mobile Agent Based PVC Configuration Management	69
Figure 5.5 Mapping Common View to Vendor Specific Switch	72
Figure 5.6 Common View of ATM Switch	80

Figure 5.7 Simulated Switches	81
Figure 5.8 Switch Interfaces	81
Figure 5.9 PVC Netlets	82
Figure 5.10 Class Relationship of PVC Configuration Management Application	82
Figure 5.11 Message Interaction Diagram for Setting up PVC Scenario	85
Figure 5.12 Message Interaction Diagram for Setting up PVC Scenario	86
Figure 5.13 Message Interaction Diagram for Setting up PVC Scenario	87
Figure 5.14 PVConfigManager.....	89

List of Acronyms

AAL:	ATM Adaption Layer
ABR:	Available Bit Rate
ATM:	Asynchronous Transfer Mode
CBR:	Constant Bit Rate
CCITT:	International Consultative Committee on Telegraphy and Telephony
CDV:	Cell Delay Variance
CER:	Cell Error Ratio
CLR:	Cell Loss Ratio
CMIP:	Common Management Information Protocol
CMR:	Cell Misinsertion Rate
CORBA:	Common Object Request Broker Architecture
CTD:	Cell Transfer Delay
GFC:	General Flow Control
GUI:	Graphical User Interface
HEC:	Header Error Control
IDL:	Interface Definition Language
IEEE:	Institute of Electrical and Electronics Engineers
IETF:	Internet Engineering Task Force
ILMI:	Integrated Layer Management Interface
ISO:	International Standards Organization
JVM:	Java Virtual Machine
MCD:	Mobile Code Daemon
MCTD:	Mean Cell Transfer Delay
MIB:	Management Information Base
NMS:	Network Management System
OID:	Object IDentifier
OSI:	Open Systems Interconnection
PT:	Payload Type
PVC:	Permanent Virtual Connection
QOS:	Quality Of Service
RFC:	Request For Comments
SCP:	Simulated Control Program
SECBR:	Severely-Error Cell Block Ratio
SMI:	Structure of Management Information
SNC:	Simulated Network Components
SNCD:	Simulated Network Components Database
SNMP:	Simple Network Management Protocol
SONET:	Synchronous Optical Network
TCP:	Transmission Control Protocol
TMN:	Telecommunication Network Management
UBR:	Unspecified Bit Rate
UNI:	User-Network Interface
VC:	Virtual Channel

VCC:	Virtual Channel Connection
VCI:	Virtual Channel Identifier
VCL:	Virtual Channel Link
VMC:	Virtual Managed Component
VP:	Virtual Path
VPC:	Virtual Path Connection
VPI:	Virtual Path Identifier
VPL:	Virtual Path Link

Chapter 1

Introduction

As telecommunication networks have become faster, more complex and more flexible, network management has consequently arisen as a key concern. With ATM's (Asynchronous Transfer Mode) increasing deployment in current networks, the need for their management is of great importance. This thesis tackles ATM configuration management, which is one of the major ATM network management functions.

ATM configuration management is concerned with setting up and releasing Permanent Virtual Connections (PVCs). Each ATM vendor provides their own way to configure a PVC for its ATM switches and other devices. There is no uniform way for the network operator to set up an end-to-end PVC in a heterogeneous ATM network.

The agents in existing network management systems tend to be monolithic, static, require substantial network resources and sometimes huge data transmission. Mobile agents do not statically reside on network devices, therefore can be created on demand and destroyed when no longer required. They are substantially smaller than those agents in current network management systems because they normally perform a single specific task.

This thesis proposes an innovative improvement to existing ATM configuration management solutions. Based upon the use of Mobile Agents and JAVA programming language, this work covers the Object-Oriented design and implementation of a PVC configuration management application in multi-vendor ATM networks.

The thesis consists of six chapters. This chapter presents the motivation for new way to

perform network configuration management functions in heterogeneous ATM environments. It also describes the layout of the entire thesis.

Chapter 2 summarizes the background knowledge required to understand this thesis. It first gives an overview of the basic concepts of ATM technology, then discusses ATM network management and PVC configuration management in particular. Finally, it briefly introduces a mobile code based network management. Those familiar with these topics can skip this entire chapter. Those familiar with ATM in general can skip Section 2.1. Those familiar with ATM in general and network management can skip Section 2.1 and Section 2.2.

Chapter 3 first analyses the requirements of PVC configuration management. Following that, it compares different PVC Management Information Bases (MIBs) in depth. It then introduces the necessity of a common view of a PVC MIB in multi-vendor ATM networks.

Chapter 4 analyses different approaches to PVC configuration management in heterogeneous ATM environments, including manager to manager based, CORBA based and Mobile Code based.

Chapter 5 discusses PVC configuration management application based upon Mobile Code in heterogeneous ATM networks. It first proposes a new way of PVC provisioning using a Mobile Code architecture. It then analyses the application software requirements, followed by a high level design and detailed design and implementation. Finally it demonstrates some PVC management using examples.

Chapter 6 summarizes the thesis work, then discusses some future work.

Chapter 2

Background Knowledge

The understanding of this thesis requires familiarity with ATM terminology and some domain-specific knowledge. This chapter provides readers with general brief background knowledge.

The chapter first gives an overview of basic concepts of ATM technology. Following that, it discusses ATM network management, PVC configuration management in particular. It then briefly introduces mobile code based network management.

Section 2.1 ATM Overview

ATM has been accepted world-wide through the CCITT as the transfer mode for Broadband-ISDN(B-ISDN) [1]. All services, including voice, video and data, share the same transmission and switching fabrics throughout the network.

In this section, brief discussions of ATM technologies are presented in terms of its cell structure, switching properties and virtual connection setup modes.

Section 2.1.1 ATM Cell Structure

ATM is a cell (packet) switching concept used in a network. It uses a fixed length cell of 53 octets - 48 octets of user data and a 5 octets header. Figure 2.1 depicts an ATM cell.

Figure 2.1 ATM Cell

In the 5 Octet (byte) header, the distribution of different fields and their functionality are as follows: [2], [3], [4]

General Flow Control (GFC)

This field occupies 4 bits and provides a point-to-multipoint connection. It allows multiple users to be connected to the same physical link, i.e, multiplexing the shared network among the cells of the various ATM connections.

Virtual Path Identifier (VPI), Virtual Channel Identifier (VCI) and Payload Type (PT)

These three fields are in coded 8 bits, 16 bits and 4 bits respectively at the User-Network Interface (UNI). At a Network-Network Interface (NNI) the GFC field is replaced by 4 additional VPI bits, which results in a VPI field of 12 bits. These three identifiers support recognition of an ATM cell on a physical transmission medium. VPI

and VCI are unique for cells belonging to the same virtual connection on a shared transmission medium. More details on these two fields can be found in Section 2.1.3. The PT field indicates whether the cell is carrying user information to be delivered transparently through the network, or special network information.

Cell Loss Priority (CLP)

This field only occupies 1 bit and indicates whether a cell has a high priority (CLP = 0) or has lower priority (CLP = 1). i.e., is subject to discarding in the network. Priorities can be assigned either on a per connection (per VPI/VCI) basis or on a per cell basis. In the first option, all cells in the VC/VP have the same priority, while in the second case, cells within a VC/VP may have different priorities.

Header Error Control (HEC)

Finally, the HEC field consists of 8 bits and is to protect the header against corruption. A polynomial coding algorithm is used to determine if there has been a bit error.

Section 2.1.2 ATM Architecture

The same seven layered architecture used in OSI is applied to ATM networks. The Protocol Reference Model (PRM) defined in CCITT [1], and shown herein as Figure 2.2, is composed of a *user plane*, a *control plane* and a *management plane*.

The user plane transports user information. Example protocols of using this plane are TCP/IP and FTP. The control plane deals with the signalling required to setup, supervise and release a connection. The upper layer protocol for this plane is Q.291. The management plane provides two types of functions. Firstly plane management performs manage-

ment functions related to a system as a whole and provides co-ordination between all planes. Examples of this are Simple Network Management Protocol (SNMP) and Common Management Information Protocol (CMIP). Secondly, layer management handles the management functions associated with a particular layer and also the flow of operation, administration and maintenance (OAM) information to a specific layer.

Figure 2.2 B-ISDN protocol reference model

For each plane, a layered approach is used with an independence between the layers. The physical layer supports pure medium-dependent bit functions and converts the ATM cells stream into bits to be transported over the physical medium. The most commonly used structure for this layer is SONET (Synchronous Optical NETWORK). The main functions of the ATM layer are multiplexing/demultiplexing of cells of different connections into a single cell stream on a physical layer, translation of the cell identifier and implementation of a flow control mechanism at the user-network interface (UNI). The ATM adaptation layer (AAL) enhances the service provided by the ATM layer. It performs functions

for the user, control and management planes and supports the mapping between the ATM layer and the next higher layer. To accommodate various services, several types of AAL have been defined. These services are described in detail in the next section.

Section 2.1.3 VP and VC Switching

In ATM, the transport network functions are split into two parts, namely physical layer transport functions and ATM layer transport functions. This section concentrates on those at the ATM layer [4].

The ATM layer has two hierarchical levels, the virtual channel level and the virtual path level, which are both defined in the CCITT Recommendation [1]. *Virtual channel* (VC) is 'A concept used to describe unidirectional transport of ATM cells associated by a common unique identifier value.' This identifier is called the virtual channel identifier (VCI) and is part of the cell header (see Figure 2.1). *Virtual path* (VP) is 'A concept used to describe unidirectional transport of cells belonging to virtual channels that are associated by a common identifier value.' This identifier is called the virtual path identifier (VPI) and is also part of the cell header (see Figure 2.1). The VP concept allows grouping of several virtual channels.

Concerning the VC and the VP levels of the ATM layer, it is helpful to distinguish between links and connections. A *virtual channel link* is 'A means of unidirectional transport of ATM cells between a point where a VCI value is assigned and the point where that value is translated or removed.' Similarly, a *virtual path link* is terminated by the points where a VPI value is assigned and translated or removed. A concatenation of VC links is

called a *virtual channel connection* (VCC), and likewise, a concatenation of VP links is called a *virtual path connection* (VPC). A VCC may consist of several concatenated VC links, each of which is VPC. The VPCs usually consist of several concatenated VP links. Figure 2.3 illustrates the concept of links and connections.

Figure 2.3 ATM Switches: Virtual Path and Virtual Channel Switches

VCI and VPIs in general only have significance for one link [4]. In a VCC/VPC the VCI/VPI value will be translated at VC/VP switching entities. VP switches (see Figure 2.3) terminate VP links and therefore have to translate incoming VPIs to the corresponding outgoing VPIs according to the destination of the VP connection. VCI values remain unchanged.

Figure 2.4 Virtual path switching

VC switches (similar to Figure 2.4) terminate VC links and possibly VP links. Both VPI and VCI translation is performed.

VCCs/VPCs can be employed between a user and a user, a user and a network and a network and a network. All cells associated with an individual VCC/CPC are transported along the same route through the network. Cell sequence is preserved for all VCCs.

ATM operates in a connection-oriented mode. Before cells are transmitted from one user terminal to another user terminal, a logical/virtual connection setup phase must allow the network to perform a reservation of the necessary resources, for instance, bandwidth. After cell transmission, the allocated resources are released. There are two kinds of mechanisms to set up a connection, namely Permanent Virtual Circuit (PVC) and Switched Virtual Circuit. The former is pre-established manually via a network management station at each switch along the end-to-end path. Telnet in the user plane or SNMP/CMIP in the management plane may be used to perform such tasks (see Figure 2.2). The latter is set up

on demand, based upon signaling procedures like Q.293. The control panel in Figure 2.2 is particularly designed for this purpose.

Section 2.1.4 Quality of Service and ATM Service Classes

When providing a service to a customer, the Quality of Service (QoS) needs to be considered. ISO standard defines QoS as a concept for specifying how “good” the offered networking services are. QoS can be characterized by a number of parameters. The ATM QoS features are mapped into a set of cell transfer performance parameters which correspond to the generic criteria of the assessments shown in Table 2.1[5]:

PARAMETER	ASSESSMENT
Cell Error Ratio(CER)	Accuracy
Severely-Error Cell Block Ratio(SECBR)	Accuracy
Cell Loss Ratio(CLR)	Dependability
Cell Misinsertion Rate(CMR)	Accuracy
Cell Transfer Delay(CTD)	Speed
Mean Cell Transfer Delay(MCTD)	Speed
Cell Delay Variance(CDV)	Speed

Table 2.1 ATM QoS parameters

The above parameters are based upon an ATM connection. The CER is defined as a number of errored cells divided by the total number of cells transferred during a given connection. The CLR is defined as number of lost cells divided by the total number of transmitted cells for the connection. The CMR is defined as the number of misinserted cells divided by the time interval for the connection. The SECBR is defined as the number of

severely errored cell blocks divided by the total number of transmitted blocks. A cell block is a sequence of cells transmitted consecutively on a given connection, while a severely errored cell block outcome occurs when more than a certain number of errored cells, lost cells or misinserted cells are observed in a received cell block. The SECBR can be interpreted as the probability that an ATM cell is discarded. The CTD is defined as the elapsed time between a cell exit event at one measurement point and the corresponding cell entry event at a second measurement point for the connection. The MCTD is defined as the statistical average of a specified number of cell transfer delays for one or more connections. The CDV is sometimes called delay "jitter". Knowing the MCTD and the CDV, the distribution of delays to the ATM cells in a connection can be determined.

It is sometimes difficult to accurately specify a QoS, as all of the parameters of the traffic to be supplied to the network from the customer's premises are not always definable. It has been shown in some research [6] that whilst the long term QoS of the overall network can be predicted, there are so many factors which influence the QoS in the short term that the network cannot always accurately predict the QoS offered to a customer for each single connection. It is therefore likely that the contract between the customer and network operator will specify a "best-effort" QoS without providing an absolute guarantee. A specific QoS class provides a quality of service to an ATM virtual connection (VCC or VPC) in terms of a subset of the ATM performance parameters previously defined in previous page. The ATM Forum[5] currently defines five different service classes. These account for the variety of traffic types which are intended to be carried over ATM networks. The services offered by the network differ by category. These differences are

reflected in QoS parameters and a brief discussion of each service category and its QoS is given below:

Constant Bit Rate Service (CBR)

The CBR service is intended for real-time applications which require a fixed bandwidth in order to guarantee a minimum end-to-end transmission delay of the cells. The QoS parameters required for this service class are peak-to-peak CTD, maximum CTD and CLR. This service is categorized as Class A and handled by the ATM adaptation layer AAL1 by CCITT Recommendation.

Real-time Variable Bit Rate Service (rt-VBR)

The rt-VBR service is similar to the CBR service in that it is also intended for real-time traffic. Here, however, the traffic is typically more 'bursty' in nature. The QoS parameters required for this service class are peak-to-peak CTD, maximum CTD and CLR. This service is categorized as Class B and handled by ATM adaptation layer AAL2 by CCITT Recommendation.

Non real-time Variable Bit Rate Service (nrt-VBR)

This service category is the first of the non real-time service categories. It is intended for bursty traffic types where cell loss is still not tolerated. The QoS parameters required for this service class are mean CTD, CLR, SECBR and CMR. This service is categorized as Class C and handled by ATM adaptation layer AAL3/4 by CCITT Recommendation.

Unspecified Bit Rate Service (UBR)

The UBR service category is intended for traditional traffic types such as email or file

transfer applications. This service is categorized as Class D and handled by ATM adaptation layer AAL3/4 by CCITT Recommendation.

Available Bit Rate Service (ABS)

For traffic subscribing to the ABR service class, the network guarantees to use the available bandwidth for transmission. The ATM Forum does not specify a corresponding AAL for this category. The “best-effort” QoS may be used for this category.

Section 2.2 ATM Network Management

As the communication networks increase in complexity and in size, it becomes essential to use automated network management tools to assist network administrators or operators in monitoring and control of network elements. System network management of ATM covers five subareas, which are discussed in Section 2.2.2. One of the system network management deals with the configuration of an ATM network consisting of one or more switches. In this section, an overview of the network management framework is followed by a brief discussion on functional areas of network management, management protocols and management information bases (MIBs).

Section 2.2.1 Network Management Framework

A network management system consists of four parts: a management station (or manager), an agent, a management information base (MIB), and network management protocols. Figure 2. illustrates the relationship among these parts:

Figure 2.5 Manager/Agent Model

The management station serves as an interface for the administrator with the network management system. It translates administrator's commands into actual monitoring and control of the network elements. Various services provided at the management station include applications for end-to-end PVC provisioning.

Each node in the network that participates in network management contains agent software for performing network management-related tasks, for instance, setting up a cross-connection at an ATM switch. Each agent collects data on resources and stores statistics locally. Agents also respond to commands from the manager.

The third component of the network framework is the MIB. An MIB is a collection of objects, each representing a particular aspect of a managed device. For example, a manager performs monitoring function by retrieving the value of available bandwidth at certain port of a switch, and changes the settings of the switch resource by modifying the

value of available bandwidth.

Finally, the communication between the manager and agents is carried out using a network management protocol. The two standard network management protocols are the IETF's simple network management protocol (SNMP) and ISO's common management information protocol (CMIP). In addition to defining a specific protocol, both SNMP and CMIP define a set of MIBs.

Section 2.2.2 Functional Areas of Network Management

ISO defined the following five functional areas for network management, known as FCAPS [7]:

1. Fault management:

The facilities that enable the detection, isolation, and correction of abnormal operation of network resources;

2. Performance management:

The facilities needed to evaluate the behavior of managed objects and effectiveness of communication activities;

3. Security management:

Addresses those aspects essential to operating network management system correctly and to protecting managed objects;

4. Accounting management:

The facilities that enable charges to be established for the use of managed objects and costs to be identified for the use of those managed objects;

5. Configuration management:

The facilities that exercise control over, identify, collect data from, and provide data to managed objects for the purpose of assisting in providing for continuous operation of interconnection services.

Fault management is the process of locating problems or faults in the network. It is a collection of activities required to maintain the desired network service level. Performance management measures the performance of the network hardware, software, and transmission media. It is a quantitative investigation of the network resources to verify that the service levels are maintained and to support other network management functions such as configuration and fault management.

Security management is the process of controlling network access and protecting objects within the network. It involves the protection of the transfer of authentication from one location to another. Accounting management deals with tracking the usage of the network resources by network users and the cost incurred for the service. It is also used to limit the amount of resources allocated to users and informing them of the costs incurred and resources used.

Configuration management is the process of finding and setting up network devices and their resources. It is a set of short and medium range activities for controlling network resource inventory, tracking vendor performance, maintaining trouble files, evaluating and negotiating service levels, managing and changing security levels, and dealing with costs and charging.

Applied to an ATM network including end stations, the configuration management

requirements include:

- creating and deleting ATM connections (VPCs and VCCs);
- getting connection status;
- determining the number of connections active on an interface;
- determine the maximum number of connections supported on an interface;
- determine the number of preconfigured connections on an interface;
- configuring the number of VPI/VCI bits supported.

End-to-end PVC configuration management, which is part of ATM configuration management, is the prime research area for this thesis. For more detailed PVC configuration management, see Chapter 3 PVC Configuration Management for ATM Networks.

Section 2.2.3 Network Management Protocols

The manager/agent model presented in Section 2.2.1 performs network management through the communication between a manager and several agents. The manager controls the overall network management, while the agent adjusts and controls managed objects belonging to it as directed by its manager and reports results back to the manager.

Network management requires communication protocol to exchange information between a manager and the agents. A management protocol provides a means of communication by providing several functions:

- Reading and updating attributes of managed objects;
- Requiring managed objects to perform a specific function;
- Reporting results produced by managed objects;
- Creating and deleting managed objects.

The two standard management protocols SNMP and CMIP have their own formats and

syntax to specify the managed objects, and both employ the manager/agent model. Besides these two standard protocols, some vendors developed their own management protocols for their own devices. The following summarizes the two standard management protocols:

Simple Network Management Protocol (SNMP):

SNMP was developed by IETF and is widely used to manage TCP/IP networks. It is a connectionless (UDP/IP) protocol. The SNMP agents reside on the managed devices and are designed to operate using minimal system resources, not to slow down the operation of the device significantly from providing network management services. The agents gather data about a device, which can be either static like system information, or dynamic like traffic packets, and stores them in the MIB residing on that device. The processing of the collected data is done by the management application. The management application communicates with managed devices via a set of SNMP commands. There are many commercial products based upon on SNMP.

Common Management Information Protocol (CMIP):

CMIP was developed by ISO. Unlike SNMP, CMIP is designed to provide generic solutions to overall network management. It is, however, a sophisticated protocol and is used mainly in service provider [8] networks. While SNMP is based upon a simple common structure, CMIP includes more powerful and difficult-to-develop commands. Accordingly, although CMIP offers more versatility, it is not yet as popular as SNMP. In particular, currently there are few commercial applications available that use CMIP.

Section 2.2.4 Management Information Base

The information gathered by the agents is the most important part of the management system, since all other activities are based upon it. Thus, much effort has been devoted to defining and standardizing exactly what information is to be gathered and how it is to be stored.

MIB standards define network management variables and their meanings. The variables themselves are based upon the structure of management information (SMI). The SMI specifies that the MIB variables be defined and referenced using the ISO abstract syntax notation 1 (ASN.1).

The contents of the SNMP MIB are defined using the SMI, while its structure is defined by the ISO as branch of global object identifier name-space tree. The tree is jointly administered by ISO and ITU-T. Each node in the tree has a name and a number, both of which are unique at that level of particular branch. Through a hierarchical design, the name-space tree allows individual groups authority over certain branches, as illustrated in Figure 2.6. By following a path from the root to the object, one can determine the globally unique name for that object.

Figure 2.6 Object Identifier Tree

Due to the nature of design differences on each ATM device, each vendor develops its own device MIBs. For instance, Fore has its own MIB for their ATM switches products, while Cisco has their own MIB for their ATM switches. For more detailed Fore switch MIB and Cisco switch MIB, see Chapter 3 PVC Configuration Management for ATM networks.

Section 2.3 Mobile Code Based Network Management

In the previous section, we overviewed the current network management methodologies. These network management systems tend to be monolithic, require substantial network resources and sometimes huge data transmission over the network. Therefore, they result

in difficulty in maintaining the network and the possibility of causing a bottleneck. In order to overcome these problems, a new technology for network management is developed. The Perpetuum Mobile Procura Project [9], which consists of infrastructure, simulator, network manager and application subprojects, is used to research the use of mobile code for managing networks. Due to its platform independence, mobility, portability, networking capability and other features, the JAVA programming language has been chosen for the implementation.

This section first introduces the fundamental parts of advanced network management based upon mobile code. It then discusses the feasibility of building a PVC Configuration application using the mobile code.

Section 2.3.1 Motivation for a New Approach to Network Management

With the growth of the telecommunication industry, networks have become more and more heterogeneous. A large network is made of devices by different vendors. The software used for managing such a network is even more diverse. To maintain such networks requires attention to compatibility and interoperability.

As mentioned in the last section, current network management paradigm is built on the client/server model of distributed systems. Applying the client/server model to network management, agents act as servers, while a manager acts as a client. Agents collect data about the devices and store them in a database but have very limited computational capabilities. In addition, agents wait for requests from the manager and reply to the manager. It is up to the manager to analyze the data sent by agents. That sometimes causes the bottle-

neck on a manager due to big chunks of data dumped to it. Moreover, network management applications are frequently specially designed by vendors for their own devices. Therefore they tend to be monolithic and application is difficult in a multi-vendor environment.

In a typical client/server environment, there are many small clients and a large centralized server. The server normally has powerful processing capabilities. In the network management field, there are many agents running on devices (like routers, switches) and a centralized management work station (manager). This contradicts the traditional client/server model.

To ease the computation burden on the manager side, a large management task needs to be divided into a set of small, manageable tasks. A new approach to network management based on mobile code is used to tackle the issues. The research of the Perpetuum Mobile Procura Project uses mobile code, or mobile agents [10]. With this new approach, agents no longer statically reside on devices. They are mobile, and can be created on demand. They can also be destroyed when they are no longer required. These agents tend to perform specific tasks, therefore they are normally fairly small compared to an agent in a legacy network management system. Furthermore, they can travel from one node to another through a network on a predefined migration pattern. This is advantageous to reduce the network traffic. For instance, in the legacy approach, to perform a management task involving three nodes in a network, the manager has to send a request to and receive a response from the three nodes, while in the new approach, the manager only needs to send a netlet out. The netlet will perform the task on each node on behalf of the manager and

report back the result if necessary. In terms of the number of messages required, the former requires six, while the latter only needs four.

Section 2.3.2 Framework of Network Management with Mobile Code

Currently, there is much ongoing research using mobile agents by companies such as Crystaliz, Inc., General Magic, Inc., GMD FOKUS and International Business Machines Corporation. In order to standardize mobile agents. OMG (Object Management Group) proposed the Mobile Agent Facility Specification [25]. There are also a few applications using mobile agents to do things like providing management functionality where needed [26].

To perform network management tasks with mobile code, it is necessary to have an appropriate infrastructure. Also, to test the design of the mobile code, it is reasonable to use a simulator for the test bed. This subsection introduces the basic components in the infrastructure, followed by a discussion of the simulator.

Section 2.3.2.1 Infrastructure of Mobile Code

A facility to transport portable code is a core requirement for solutions based upon mobile code [9]. The infrastructure of mobile code requires JAVA Virtual Machines (JVMs) [11] running in network components. JVM is a platform dependent interpreter, which translates JAVA code into machine code. At the time of writing, there are also JVM chips available. Figure 2.3. shows the key components required to send mobile code between running network components. The following discusses these key components [12].

Figure 2.7 Infrastructure of Mobile Code

The MCD is the core component to realize code mobility in order to perform network management tasks. It provides a set of services that enable mobile code execution. These services include: a runtime environment, a migration facility, and an interface to access managed resources.

The MCD is a thread running inside the JVM and listens at two communication ports to accept mobile code. One connection port is TCP, while the other is UDP. Once a piece of mobile code, for instance, a netlet as a mobile agent, is accepted, it is instantiated within the same JVM as MCD is running. The MCD also keeps track of all handles of instantiated mobile code.

The migration facility is a part of MCD, which is responsible for transporting a mobile agent to the next location. The migration route can be decided by the application manager, the MCD, or the agent itself. When an agent's migration is requested by an application manager, another agent, or the agent itself, the MCD calls methods `onMigrate()`. This notifies the agent to be transported to the next location. Therefore the agent will have time to finish the task. When the migration is completed, the agent is destroyed and removed from the MCD.

The ability to access managed resources of a network component is a fundamental in network management. Accessing data in a heterogeneous system may be complex due to non-standard naming schema and procedures. Therefore, there must be a uniform way to be understood by the mobile code in performing its tasks without prior knowledge of the underlying system. The VMC interface makes it possible for mobile agents to access managed resources. The VMC acts as bridge between a mobile agent and managed resources.

Section 2.3.2.2 Network Simulator

It is not realistic to expect to test the mobile code in a real network. Therefore a simulator is necessary to test and experiment the ideas and designs. The simulator subproject [13] is part of the Perpetuum Mobile Procura project. From the perspective of an agent, there is no difference between a simulated and real network component as long as they run MCD that is able to accept and execute netlet code.

The simulator, also known as a simulated network environment, is made of the simulate control program (SCP) and simulated network components (SNCs). Figure 2.4 shows

the network architecture.

Figure 2.8 Network Simulator Architecture

The SCP is composed of: SCP applet thread, SCP dispatcher thread and the SNC database. The SCP applet presents a graphical user interface to the human user. It takes user input and displays the corresponding results. It also allows the user to manipulate SNC registration information in the SNCD. The user interface has the following functionality:

- create SNC;
- manage SNC links;
- destroy SNC;
- ship code to SNC;
- modify SNC MIB data

The SNCD is the central component of SCP, which contains all SNC registration infor-

mation and link status. Each SNC has fields real IP/Port and simulated IP/Port. The former is used for communications between SCP and SNC, while the latter is used to simulate a real network component. The relationship of the two IP/Port can be found in the SNCD.

The Dispatcher thread is the component that handles communication between the SCP and SNCs. It is spawned by SCP at start-up time and listens for communication requests from SNCs at a port. The communication is conducted by SCPCommunicator through a TCP socket.

A simulated network component (SNC) is a collection of JAVA objects running as a thread in JVM. Its purpose is to simulate a real network component such as an ATM switch. It is composed of an MCD, a VMC or sometimes a netlet. The VMC thread has a handle of managed sources, or a MIB object.

Section 2.3.3 PVC Configuration Application Based on Mobile Code

Setting up an end-to-end Permanent Virtual Connection in an ATM network is not an easy task. It involves several parties in the route. When the network is composed of multi-vendor switches, the task becomes even more complex due to the heterogeneous nature of vendor dependent devices.

With the mobile code mentioned in the previous subsection, simplifying PVC configuration management becomes feasible. Although different switches have their own ways to store data, they all perform more or less similar kinds of tasks in terms of PVC configuration. Using mobile code, an operator no longer needs to have the knowledge of a specific

switch. All he or she has to know is the common representation of the PVC subset of the switch MIB. Upon request to set up an end-to-end PVC, a specialized netlet, namely PVC-Netlet is dispatched to each switch to perform the PVC configuration. At each vendor-specific switch, there is a VMC which maps the common representation held by the netlet to the specific switch MIB. The PVCNetlet traverse all the switches in the path.

Due to the difficulty of performing the experiment on a real ATM network, the test was conducted in a simulated ATM environment. For more detail of the PVC configuration application based upon mobile code, see Chapter 5.

Chapter 3

ATM PVC Configuration Management

As one of the ATM network management functions, ATM configuration management is concerned with setting up and releasing the Permanent Virtual Connection (PVC). Due to the connection nature of ATM, before sending traffic cells, the proper virtual circuit needs to be configured. After the virtual circuit is set up, the traffic cells will then follow the same route.

This chapter first analyses the requirement of PVC configuration. It then compares different PVC MIBs in depth. Lastly it introduces the necessity of the common view of a PVC MIB.

Section 3.1 PVC Configuration Requirements

ISO defined the five functional areas for network management: Fault management, Configuration management, Accounting management, Performance management and Security management (FCAPS). The main objective of a network management system is to provide the means for communication systems in all nodes connected to a network to be monitored and controlled. That is, an important goal is to support an integrated approach to the management of a network which contains multi-vendor devices, software packages and carriers.

A configuration management system must have some functions related to resources (management objects), for example, adding (removing) a resource, such as a switch to (from) a network, initializing a resource and moving a resource around. A configuration

management system must allow an user to be able to allocate a managed resource, such as bandwidth.

Among the five network management functions, the PVC management is considered as configuration management function. The requirements for PVC management include:

- creating and deleting ATM connections (VPCs and VCCs);
- getting connection status;
- determining the number of connections active on an interface;
- determining maximum number of connections supported on an interface;
- determining the number of preconfigured connections on an interface;
- configuring the number of VPI/VCI bits supported.

Section 3.2 PVC MIBs

One of the key components of the network management framework is the MIB [8]. A MIB is a collection of objects, each representing a particular aspect of a managed agent. In ATM networks each switch has an agent process (agent daemon) running and a corresponding switch MIB. A manager can perform a monitoring function by retrieving the value of a particular object, for instance, tracing a PVC. As well it can change the settings of a network resource by modifying the value of the corresponding object, for instance, setting up a PVC.

The communication between the manager and agents is carried out using a network management protocol. The two standard network management protocols are the IETF's simple network management protocol (SNMP) and ISO's common management information protocol (CMIP). Each protocol defines a corresponding database structure specification and a set of data objects (MIBs). This thesis concentrates on SNMP. The specification

used in SNMP is Abstract Syntax Notation 1 (ASN.1). Besides these two standard network management protocol, some switch vendors employ their proprietary protocols and corresponding MIBs.

The contents of the SNMP MIB are defined using the Structure of Management Information (SMI), while its structure is defined by ISO as a branch of the global object identifier (OID) name-space tree. This tree is jointly administered by ISO and ITU-T. Each node in the tree has a name and a number, both of which are unique at that level of the particular branch of the tree. Through a hierarchical design, the OID tree allows individual groups authority over certain branches, as illustrated in Figure 2.6. By following a path from the root to the object, one can determine the global name for that object.

This chapter compares four SNMP ATM MIBs in the context of PVC configuration. The first two are standard, namely Integrated Layer Management Interface 4.0 (ILMI) by the ATM Forum and RFC 1695 ATOM by Internet Engineering Task Force (IETF). The other two are vendor specific by Fore and Cisco.

Section 3.2.1 ATM Forum ILML 4.0 MIB (UNI 3.1)

This section and the next section describe those managed objects from two standard MIBs with respect to configuring a PVC.

While B-ISDN ATM is a definition for public networks, it can also be used within private networking products. In recognition of this fact, the ATM Forum defines two distinct forms of User to Network Interface (UNI) [14]:

1. Public UNI - which is typically used to interconnect an ATM user with an ATM
-

switch deployed in a public service provider's network,

2. Private UNI - which is typically used to interconnect an ATM user with an ATM switch that is managed as part of the same corporate network.

The ATM Forum's ILMI covers the both UNI interfaces. The primary distinction between these two classes of UNI is physical reach. There are also some functionality differences between the public and private UNI due to the applicable requirements associated with each of these interfaces. Both UNIs share an ATM layer specification, but may utilize different physical media. The term "ATM user" represents any device that makes use of an ATM network, via an ATM UNI.

Each UNI has a UNI management entity (UME), which maintains the status information of the particular UNI and responds to SNMP messages. The different NMS on either side of the UNI use a pre-specified virtual circuit to communicate SNMP messages which are encapsulated into ATM (ILMI) cells using AAL5. Its VC has the values VPI = 0 and VCI = 16. The types of managed objects are depicted in Figure 3.1:

*denotes PVC configuration related group

Figure 3.1 ATM Forum UNI ILMI MIB

The following tables provide a summary of the different managed objects contained in the ILMI 4.0 MIB [15] in the context of PVC management. Table 3.2.1.1 describes those managed objects in the Physical Port Group.

Object	Syntax	Access	Status	Description
atmPort Table	SEQUENCE of atmPortEntry	NA	M	A list of port entries
atmPort Entry	SEQUENCE	NA	M	A port entry contains information about the physical layer of an ATM interface
atmPort Index	INTEGER	RO	M	A unique value for each port entry
atmPort Address	AtmAddress	RO	O	The ATM address for the port
atmPortMy IfName	DisplayString	RO	M	Identical with ifName in physical interface group from MIBII

atmMyIpAddress	IPAddress	RO	M	An IP address to which a NMS can send SNMP messages to UDP port 161
----------------	-----------	----	---	---

Table 3.2.1.1 Physical Port Group

Table 3.2.1.2 describes those managed objects in the ATM Layer Group.

Object	Syntax	Access	Status	Description
atmLayer Table	SEQUENCE of atmLayer Entry	NA	M	A list of ATM Layer entries
atmLayer Entry	SEQUENCE	NA	M	An entry contains information about the ATM layer of an ATM interface
atmLayer Index	INTEGER	RO	M	A unique value for each ATM Layer entry
atmLayer MaxVPCs	INTEGER (0...4096)	RO	M	The maximum permanent VPCs supported on this ATM interface
atmLayer MaxVCCs	INTEGER (0...268435456)	RO	M	The maximum permanent VCCs supported on this ATM interface
atmLayer Configured VPCs	INTEGER (0...4096)	RO	M	The number of permanent VPCs configured on this ATM interface
atmLayer-ConfiguredVCCs	INTEGER (0...268435456)	RO	M	The number of permanent VPCs configured on this ATM interface
atmLayer Device Type	INTEGER	RO	M	The type of the ATM device, end user takes user(1) and network node take node(2)

Table 3.2.1.2 ATM LayerGroup

For the purpose of simplifying PVC connections, all QoS parameters are not considered and only UBR type traffic is transmitted through the network. Table 3.2.1.3 describes those managed objects in the Virtual Path Connection Group in the context of PVC configuration.

Object	Syntax	Access	Status	Description
atmVpc Table	SEQUENCE of atmVpc Entry	NA	M	A list of ATM VPC entries
atmVpc Entry	SEQUENCE	NA	M	An entry contains information about a particular virtual path connection
atmVpc PortIndex	INTEGER	RO	M	The corresponding physical port index for a particular VPC connection
atmVpc Vpi	INTEGER (0...4096)	RO	M	The VPI value of this Circuit Path Connection
atmVpc OperStatus	INTEGER	RO	M	The present operational status of the VPC, values can be unknown(1), end2endUp(2), end2endDown(3), localUpend2endUnknown(4) and localDown(5)
atmVpc BestEffort Indicator	TruthValue	RO	M	If the value is true(1) the network is required to apply UBR conformance

Table 3.2.1.3 Virtual Path Connection Group

Table 3.2.1.4 describes those managed objects in the Virtual Channel Connection Group in the context of PVC configuration.

Object	Syntax	Access	Status	Description
atmVcc Table	SEQUENCE of atmVcc Entry	NA	M	A list of ATM VCC entries
atmVcc Entry	SEQUENCE	NA	M	An entry contains information about a particular virtual channel connection
atmVcc PortIndex	INTEGER	RO	M	The corresponding physical port index for a particular VCC connection

atmVpc Vpi	INTEGER (0...4096)	RO	M	The VPI value of this Virtual Channel Connection
atmVcc Vci	INTEGER (0...65535)	RO	M	The VCI value of this Virtual Channel Connection
atmVpc OperStatus	INTEGER	RO	M	The present operational status of the VPC, values can be unknown(1), end2endUp(2), end2endDown(3), localUpend2endUnknown(4) and localDown(5)

Table 3.2.1.4 Virtual Channel Connection Group

One noticeable observation from ILMI 4.0 MIB is that all variables are Read Only. This is due to lack of security of SNMP v1.

Section 3.2.2 IETF ATM MIB (RFC 1695)

RFC 1695 defines a MIB used for managing ATM-based interfaces, devices, networks and services. Its primary goal is to manage ATM PVCs [16]. The grouped objects are illustrated in Figure 3.2.

Figure 3.2 IETF ATM MIB

The following tables provide a summary of the different managed objects contained in the ATM MIB in the context of PVC management. Since this MIB uses SNMP v2 the headings are slightly different from the ILMI MIB, which uses SNMP v1. Table 3.2.2.1 describes those managed objects in the Interface Configuration Parameters Group. Notice that this group is similar to the Physical Interface group and ATM Layer group from the ILMI MIB.

Object	Syntax	Max-Access	Status	Description
interfaceConfTable	SEQUENCE of inetrfConfEntry	NA	C	A list of ATM local interface parameters, one entry per interface port
interfaceConfEntry	SEQUENCE	NA	C	An entry contains information about the ATM interface, indexed by ifIndex in physical interface group of MIB II
interfaceMaxVpcs	INTEGER (0..4096)	RW	C	Maximum number of vpcs (PVCs) supported at this ATM interface
interfaceMaxVccs	INTEGER (0..65536)	RW	C	Maximum number of vccs (PVCs) supported at this ATM interface
interfaceConfVpcs	INTEGER (0..4096)	RO	C	The number of configured vpcs (PVCs) at this ATM interface
interfaceConfVccs	INTEGER (0..65536)	RO	C	The number of configured vccs (PVCs) at this ATM interface
interfaceAddressType	INTERGER	RO	C	The type of ATM address configured for use at this interface
interfaceMyNeighbourIpAddress	IpAddress	RW	C	This value may be obtained either through manual configuration or through ILMI interaction with the neighbor system

interfaceMy Neighbour IfName	DisplayString	RW	C	This value may be obtained either through manual configuration or through ILMI interaction with the neighbor system
------------------------------------	---------------	----	---	---

Table 3.2.2.1 Interface Configuration Parameters Group

An ATM Virtual Path Link (VPL) contains configuration and state information of a list of bi-directional VPLs. A VPL is terminated in an ATM host or switch and can be created, deleted and modified. A VPL entry is indexed by ifIndex and vplVpi. Table 3.2.2.2 describes those managed objects in the VPL Group. Notice that this group has the similarity with Virtual Path Connection group from the ILMI MIB.

Object	Syntax	Max-Access	Status	Description
vpl Table	SEQUENCE of vplEntry	NA	C	A list of ATM VPLs, one bi-directional VPL as one entry
vpl Entry	SEQUENCE	NA	C	An entry contains information about the ATM VPL information
vpl Vpi	INTEGER (0..4096)	NA	C	The VPI value of this VPL
vplOper Status	INTEGER	RO	C	The current operational status of the VPL, up(1) down(2), unknown(3)
vplLast Change	TimeStamp	RO	C	The value of MIB II's sysUpTime object at the time this VPL entered its current operational status
vplRow Status	RowStatus	RC	C	This object is used to create, delete or modify a VPL

Table 3.2.2.2 Interface Virtual Path Link Group

Similar to ATM VPL, an ATM Virtual Channel Link (VCL) contains configuration and state information of a list of bi-directional VCLS. A VCL is terminated in an ATM

host or switch and can be created, deleted and modified. A VCL entry is indexed by *ifIndex* , *vplVpi* and *vclVci*. Table 3.2.2.3 describes those managed objects in the VCL Group. Notice that this group is similar to the Virtual Channel Connection group from ILMI MIB.

Object	Syntax	Max-Access	Status	Description
vcl Table	SEQUENCE of vclEntry	NA	C	A list of ATM VCLs, one bi-directional VCL as one entry
vcl Entry	SEQUENCE	NA	C	An entry contains information about the ATM VCL information
vcl Vpi	INTEGER (0..4096)	NA	C	The VPI value of this VCL
vcl Vci	INTEGER (0..65535)	NA	C	The VCI value of this VCL
vclOper Status	INTEGER	RO	C	The current operational status of the VPL, up(1) down(2), unknown(3)
vplLast Change	TimeStamp	RO	C	The value of MIB II's sysUpTime object at the time this VCL entered its current operational status
vplRow Status	RowStatus	RC	C	This object is used to create, delete or modify a VCL

Table 3.2.2.3 Interface Virtual Connection Link Group

An ATM Virtual Path (VP) Cross Connection Group contains configuration and state information of a list of all point-to-point, point-to-multipoint, or multipoint-to-multipoint VP cross connections. For illustration purposes, only point-to-point PVC connections are considered. A VP is cross connected at a switch or a network and can be created, deleted and modified. The term Low and High corresponds incoming interface and outgoing interface associated with a VPC cross connection. A VP entry is indexed by

vpCrossConnecIndex, vpCrossConnecLowIfIndex, vpCrossConnecLowVpi, vpCrossConnecHighIfIndex and vpCrossConnecHighVpi. Table 3.2.2.4 describes those managed objects in the VP Group.

Object	Syntax	Max-Access	Status	Description
vpCross-ConnectIndexNext	INTEGER (0...214783647)	RO	C	*This object contains an appropriate value to be used for vpCrossConnectTable.
vpCross Connect Table	SEQUENCE of vpCrossConnectEntry	NA	C	A bi-directional VP cross connection which cross-connects two VPLs is modeled as one entry
vpCrossConnectEntry	SEQUENCE	NA	C	An entry contains information about the ATM VP cross connection information
vpCrossConnectIndex	INTEGER (0...214783647)	NA	C	The unique value to identify this cross connection
vpCrossConnectLowIfIndex	IfIndex	NA	C	The value of this object is equal to MIB II's incoming ifIndex value of the ATM interface port for this cross connection
vpCross Connect LowVpi	INTEGER (0...4096)	NA	C	The value of this object is equal to VPI value at the incoming ATM interface with the VP cross connection
vpCrossConnectHighIfIndex	IfIndex	NA	C	The value of this object is equal to MIB II's outgoing ifIndex value of the ATM interface port for this cross connection
vpCross Connect HighVpi	INTEGER (0...4096)	NA	C	The value of this object is equal to VPI value at the outgoing ATM interface with the VP cross connection

vpCrossConnectL2H OperStatus	INTEGER	RO	C	The current operational status of the VP cross connect from low to high direction, up(1) down(2), unknown(3)
vpCrossConnectH2L OperStatus	INTEGER	RO	C	The current operational status of the VP cross connect from high to low direction, up(1) down(2), unknown(3)
vpCrossConnectL2H LastChange	TimeStamp	RO	C	The value of MIB II's sysUpTime object at the time this VP entered its current operational status from low to high direction
vpCrossConnectH2L LastChange	TimeStamp	RO	C	The value of MIB II's sysUpTime object at the time this VP entered its current operational status from high to low direction
vpCrossConnect RowStatus	RowStatus	RC	C	This object is used to create, delete or modify a VP cross connection

Table 3.2.2.4 Virtual Path Cross Connection Group

** The value 0 indicates that no unassigned entries are available .*

To obtain the vpCrossConnectIndex value for a new entry, the manager issues a 'get' operation to obtain the current value of this object. After each retrieval, the agent should modify the value to the next unassigned index.

Similar to the ATM VP Cross Connection Group, ATM Virtual Channel (VC) Cross connection Group contains configuration and state information of a list of all point-to-point, point-to-multipoint, or multipoint-to-multipoint VC cross connections. For illustration purposes, only point-to-point VC connections are considered. A VC is cross connected at a switch or a network and can be created, deleted and modified. The term Low and High corresponds to incoming interface and outgoing interface associated with a VPC cross connection. A VC entry is indexed by vcCrossConnecIndex,

vpCrossConnecLowIfIndex, vcCrossConnecLowVpi, vcCrossConnecLowVci, vcCrossConnecHighIfIndex, vcCrossConnecHighVpi and vcCrossConnecHighVci. Table 3.2.2.5 describes those managed objects in the VC Group.

Object	Syntax	Max-Access	Status	Description
vcCross-ConnectIndexNext	INTEGET (0...214783647)	RO	C	*This object contains an appropriate value to be used for vcCrossConnectTable.
vcCross Connect Table	SEQUENCE of vcCrossConnectEntry	NA	C	A bi-directional VC cross connection which cross-connects two VCLs is modeled as one entry
vcCrossConnectEntry	SEQUENCE	NA	C	An entry contains information about the ATM VC cross connection information
vcCrossConnectIndex	INTEGET (0...214783647)	NA	C	The unique value to identify this cross connection
vcCrossConnectLowIfIndex	IfIndex	NA	C	The value of this object is equal to MIB II's incoming ifIndex value of the ATM interface port for this cross connection
vcCross Connect LowVpi	INTEGER (0...4096)	NA	C	The value of this object is equal to VPI value at the incoming ATM interface with the VC cross connection
vcCross Connect LowVci	INTEGER (0...65535)	NA	C	The value of this object is equal to VCI value at the incoming ATM interface with the VC cross connection
vcCrossConnectHighIfIndex	IfIndex	NA	C	The value of this object is equal to MIB II's outgoing ifIndex value of the ATM interface port for this cross connection

vcCrossConnectHighVpi	INTEGER (0...4096)	NA	C	The value of this object is equal to VPI value at the outgoing ATM interface with the VC cross connection
vcCrossConnectHighVci	INTEGER (0...65535)	NA	C	The value of this object is equal to VCI value at the outgoing ATM interface with the VC cross connection
vcCrossConnectL2HOperStatus	INTEGER	RO	C	The current operational status of the VC cross connect from low to high direction, up(1) down(2), unknown(3)
vcCrossConnectH2LOperStatus	INTEGER	RO	C	The current operational status of the VC cross connect from high to low direction, up(1) down(2), unknown(3)
vcCrossConnectL2HLastChange	TimeStamp	RO	C	The value of MIB II's sysUpTime object at the time this VC entered its current operational status from low to high direction
vcCrossConnectH2LLastChange	TimeStamp	RO	C	The value of MIB II's sysUpTime object at the time this VC entered its current operational status from high to low direction
vcCrossConnectRowStatus	RowStatus	RC	C	This object is used to create, delete or modify a VP cross connection

Table 3.2.2.5 Virtual Channel Cross Connection Group

* The value 0 indicates that no unassigned entries are available.

To obtain the vcCrossConnectIndex value for a new entry, the manager issues a 'get' operation to obtain the current value of this object. After each retrieval, the agent should modify the value to the next unassigned index.

Both VP cross connection and VC cross connection groups are available in the ATM MIB but are not available in the ILMI MIB since the latter only defines the UNI.

An ATM MIB is designed in such a way that a PVC can be configured through SNMP, which is not the case with ILMI MIB. The detailed procedures for configuring a PVC is described in Chapter 5. Any vendor specific switches MIBs which are ATM MIB compliant can use SNMP to configure PVC.

Section 3.2.3 Fore Runner Switch MIB

This section and the next section describe those managed objects from two vendor specific SNMP MIBs with respect to configuring a PVC.

The Fore switch MIB does not follow either ILMI MIB or ATM MIB. It has its own way to organize the managed objects using SNMP v1. Figure 3.3 depicts the tree structure of the Fore switch MIB [17].

*denotes PVC configuration related group

Figure 3.3 Fore ATM MIB

The following tables provide a summary of the different managed objects contained in the Fore switch MIB in the context of PVC management. Table 3.2.3.1 describes those managed objects in the Port Group. Each port entry is indexed by portNumber

Object	Syntax	Access	Status	Description
numberOf Ports	INTEGER	RO	M	The number of ports at this ATM switch
port Table	SEQUENCE of portEntry	NA	M	A list of port entries
port Entry	SEQUENCE	NA	M	An entry contains information about this switch port

port Number	INTEGER	RO	M	A unique value for each port entry
portOper Status	INTEGER	RO	M	The current operational status of this port., up(1), down(2)
port Time	TimeTicks	RO	M	The length of time this port has been in its current state, in hundredths of a second
portRemote AtmAddress	AtmAddress	RO	M	The ATM address of the entity connected to this port
portRemote IpAddress	IpAddress	RO	M	The IP address of the entity connected to this port
portMax PathIns	INTEGER	RO	M	The maximum number of incoming virtual paths on this port
portNum PathIns	GAUGE	RO	M	The number of incoming virtual paths on this port
portMax BandwidthIn	INTEGER	RO	M	The maximum incoming bandwidth on this port, in cells per second
portAlloc BandwidthIn	GAUGE	RO	M	The allocated incoming bandwidth on this port, in cells per second
portUsed BandwidthIn	GAUGE	RO	M	The used incoming bandwidth on this port, in cells per second
portMax PathOuts	INTEGER	RO	M	The maximum number of outgoing virtual paths on this port
portNum PathOuts	GAUGE	RO	M	The number of outgoing virtual paths on this port
portMaxBand widthOut	INTEGER	RO	M	The maximum outgoing bandwidth on this port, in cells per second
portAllocBan dwidthOut	GAUGE	RO	M	The allocated outgoing bandwidth on this port, in cells per second
portUsedBan dwidthOut	GAUGE	RO	M	The used outgoing bandwidth on this port, in cells per second

Table 3.2.3.1 Port Group

Table 3.2.3.2 describes those managed objects in the In Path Group. Each entry is indexed by inPathPort and VPI.

Object	Syntax	Access	Status	Description
inPath Table	SEQUENCE of inPathEntry	NA	M	A list of virtual path entries passing through a port
inPath Entry	SEQUENCE	NA	M	An entry contains information about this incoming path
inPathPort	INTEGER	RO	M	The value of this object identifies the input port of this path
inPathVpi	INTEGER	RO	M	The input of VPI value of this path.
inPathStatus	EntryStatus	RW	M	The status of this entry
inPathNum Outputs	GAUGER	RO	M	The number of output ports to which this path is routed
inPathMax Channels	INTEGER	RO	M	The maximum number of virtual channels that can be allocated on this path
inPathNum Channels	GAUGE	RO	M	The number of virtual channels that are currently allocated on this path
inPathMax Bandwidth	INTEGER	RO	M	The maximum bandwidth of this path, in cells per second
inPathAlloc Bandwidth	GAUGE	RO	M	The allocated bandwidth of this path, in cells per second
inPathUsed Bandwidth	GAUGE	RO	M	The used bandwidth of this path, in cells per second
inPathUp Time	TimeTicks	RO	M	The length of time this path has been in its current state, in hundredths of a second

Table 3.2.3.2 In Path Group

Table 3.2.3.3 describes those managed objects in the Path Route Group. Each entry is indexed by pathrInPort, pathrInVpi, pathrOutPort, and pathrOutVpi.

Object	Syntax	Access	Status	Description
pathRoute Table	SEQUENCE of pathRouteEntry	NA	M	A list of information about the routing of paths through a port

pathRoute Entry	SEQUENCE	NA	M	An entry contains information about this path route
pathrInPort	INTEGER	RO	M	The value of this object identifies the input port of this path
pathrInVpi	INTEGER	RO	M	The input of VPI value of this path.
pathrOut Port	INTEGER	RO	M	The value of this object identifies the output port of this path
pathrOutVpi	INTEGER	RO	M	The output of VPI value of this path.
pathrStatus	EntryStatus	RW	M	The status of this entry
pathrMax Bandwidth	INTEGER	RO	M	The maximum bandwidth of this path, in cells per second
pathrAlloc Bandwidth	GAUGE	RO	M	The allocated bandwidth of this path, in cells per second
pathrUp Time	TimeTicks	RO	M	The length of time this path has been in its current state, in hundredths of a second

Table 3.2.3.3 Path Route Group

Table 3.2.3.4 describes those managed objects in the Out Path Group which is similar to In Path Group. Each entry is indexed by outPathPort and vpi.

Object	Syntax	Access	Status	Description
outPath Table	SEQUENCE of outPathEntry	NA	M	A list of virtual path entries originating at a port
outPath Entry	SEQUENCE	NA	M	An entry contains information about this outgoing path
outPath Port	INTEGER	RO	M	The value of this object identifies the output port of this path
outPathVpi	INTEGER	RO	M	The VPI value of this path.
outPathStatus	EntryStatus	RW	M	The status of this entry
outPathMax Channels	INTEGER	RO	M	The maximum number of virtual channels that can be allocated on this path
outPathNum Channels	GAUGE	RO	M	The number of virtual channels that are currently allocated on this path

outPathMax Bandwidth	INTEGER	RO	M	The maximum bandwidth of this path, in cells per second
outPathAlloc Bandwidth	GAUGE	RO	M	The allocated bandwidth of this path, in cells per second
outPathUsed Bandwidth	GAUGE	RO	M	The used bandwidth of this path, in cells per second
outPathUp Time	TimeTicks	RO	M	The length of time this path has been in its current state, in hundredths of a second

Table 3.2.3.4 Out Path Group

Table 3.2.3.5 describes those managed objects in the Channel Route Group which is similar to Path Route group. Each entry is indexed by chanrInPort, chanrInVpi, chanrInVci, chanrOutPort, chanrOutVpi and chanrOutVci.

Object	Syntax	Access	Status	Description
chanRoute Table	SEQUENCE of chanRouteEntry	NA	M	A list of information about the routing of channels through a port
chanRoute Entry	SEQUENCE	NA	M	An entry contains information about this channel route
chanrIn Port	INTEGER	RO	M	The value of this object identifies the input port of this channel
chanrInVpi	INTEGER	RO	M	The input of VPI value of this channel
chanrInVci	INTEGER	RO	M	The input of VCI value of this channel.
chanrOut Port	INTEGER	RO	M	The value of this object identifies the output port of this channel
chanrOutVpi	INTEGER	RO	M	The output of VPI value of this channel.
chanrOutVci	INTEGER	RO	M	The output of VCI value of this channel.
chanrStatus	EntryStatus	RW	M	The status of this entry

Table 3.2.3.5 Channel Route Group

The way the Fore switch MIB is designed makes it impossible to use SNMP to configure a PVC.

Section 3.2.4 Cisco LightStream Switch MIB

The Cisco LightStream switch MIB is compliant to ATM MIB. Its MIB structure is an expansion of ATM MIB structure. Figure 3.4 depicts the tree structure of the Cisco LightStreamswitch MIB [18].

*denotes PVC configuration related group

Figure 3.4 Cisco LightStreamSwitch MIB

The following tables provide a summary of the different managed objects contained in the Cisco switch MIB in the context of PVC management. Table 3.2.4.1 describes those managed objects in the LineInterface Group. Each interface entry is indexed by linfIndex.

Object	Syntax	Access	Status	Description
linfStatus Table	SEQUENCE of linfStatusEntry	NA	M	A list of status of the line cards including ATM specific information

linfStatus Entry	SEQUENCE	NA	M	An entry contains information about a line card
linf Index	INTEGER	RO	M	A unique value for each port entry, same as ifIndex in MIB II
linfFwd Available Bandwidth	INTEGER	RO	M	The amount of bandwidth (Mbps) available/unallocated in the forward (incoming) direction
linfBwd Available Bandwidth	INTEGER	RO	M	The amount of bandwidth (Mbps) available/unallocated in the back-ward (outgoing) direction

Table 3.2.4.1 Line Interface Group

Table 3.2.4.2 describes those managed objects in the Connection Group. Each PVC connection entry is indexed by connIndex, connLowIfIndex, connLowVpi, connLowVci, connHighIfIndex, connHighVpi and connHighVci.

Object	Syntax	Access	Status	Description
connIndex Next	INTEGER	RO	M	The next available index for a row of the connTable
conn Table	SEQUENCE of connEntry	NA	M	A list of current PVCs information. The table is also used to establish and delete PVCs
conn Entry	SEQUENCE	NA	M	An entry contains information about a PVC connection
conn Index	INTEGER	RO	M	A unique value for each point-to-point connection
connLow IfIndex	INTEGER	RO	M	The line interface number (ifIndex) of the low port.
connLow Vpi	INTEGER	RO	M	The VPI value of the connection at the low port
connLow Vci	INTEGER	RO	M	The VCI value of the connection at the low port
connLow Throughput	INTEGER	RO	M	The throughput of the connection (Mbps) from the low port to high port

connHighIfIndex	INTEGER	RO	M	The line interface number (ifIndex) of the high port.
connHighVpi	INTEGER	RO	M	The VPI value of the connection at the high port
connHighVci	INTEGER	RO	M	The VCI value of the connection at the high port
connHighThroughput	INTEGER	RO	M	The throughput of the connection (Mbps) from the high port to low port
connRowStatus	INTEGER	RW	M	The status of this connection, active(1), notInService(2), notReady(3), createAndWait(4), destroy(5)
connCause	INTEGER	RO	M	The result of the PVC establishment/deletion, existing(1), vpivci-Busy(2),vpivciOutOfRange(3), rateOverflow(4).

Table 3.2.4.2 VPC Connection Group

Section 3.3 Common View of PVC MIB

The goal of setting up an end-to-end PVC based upon mobile code is to provide a uniform way for the ATM network operator to perform this operation. Thus he or she no longer needs to know each underlying switch MIB representation in a heterogeneous ATM network. In order to accomplish this, it is necessary to have a common representation, or common view of a PVC MIB.

Although different ATM standard organizations and switch vendors have their own ways to present their switch MIBs, in terms of PVC configuration functionality, a common view of a PVC MIB can be extracted. Table 3.3 summarizes the comparison of different MIBs based on previous section and the common view of PVC MIB.

Mgmt. Info	Managed Objects				
	RFC 1696	ILMI 4.0	Fore Runner	Cisco LightStream	Common View
Interface	ifIndex	portIndex	portNumber;	linfIndex;	ifIndex
	*ifMyNeighborIpAddress		portRemoteIpAddress		*ifMyNeighborIpAddress
	*ifMyNeighborIfName				
			portMaxBandwidthIn/Out;	linfFwd/BkwdAvail- ableBandwidth	ifIn/OutMaxBw;
			portAllocBandwidthIn/Out;		
Interface			portUsedBandwidthIn/Out;		
		atmLayerIndex;			
	ifMaxVpcs/Vccs;	atmLayerMaxVpcs/Vccs;	in/outPathMaxChanns;	NA	ifMaxVpcs/Vccs;
	ifConfVpcs/Vccs;	atmLayerConfVpcs/Vccs;	in/outPathNumChanns;		ifConfVpcs/Vccs;
	ifMaxActiveVpi/VciBits;	atmLayerMaxVpi/VciBits;			
VP/VC link	ifLmiVpi/Vci;				
		vpc/vccPortIndex;	in/outPathPort, chanPort;	NA	ifIn/OutNumber;
					linkIn/OutNumber;
	vplVpi/vclVpi, vclVci;	vpcVpi/vccVpi, vccVci;	in/outPathVpi, chanVpi/vci;		linkVpi/Vci;
	vpl/vclReceive/Transmit- TrafficDescrIndex;	vpc/vccReceive/Transmit- TrafficDescriptor;			
		vpc/vccBestEffortIndicator;			
			in/outPathMaxBw;		linkAvailableBw;
			in/outPathAlloc/ UsedBw;		linkUsedBw;
			chanAlloc/UsedBw;		
	**vpl/vclCrossConnectId;				
	vpl/vclRowStatus;		in/outPathStatus, chanStatus		linkRowStatus;
	vpl/vclAdminStatus				

VP/VC				
vp/vcCrossConnect- Next;	NA		pvcIndexNext;	
vp/vcCrossConnectIndex;			pvcIndex;	vp/vcXConnectIndex;
vp/vcCrossConnect- Low/HighIfIndex;		pathr/chanrIn/OutPort;	pvcLow/HighIfIndex	vp/vcXConnectIndexIn/ outIfIndex;
vp/vcCrossConnect- Low/HighVpi;		pathr/chanrIn/OutVpi;	pvcLow/HighVpi	vp/vcXConnectIndexIn/ OutVpi;
vcCrossConnectLow/ HighVci;		chanrIn/OutVci;	pvcLow/HighVci	vp/vcXConnectIndexIn/ OutVci;
		pathrMaxBandwidth;		vp/vcXConnectIndexIn/ OutAvailableBw;
		pathrAllocBandwidth;	pvcLow/highThroughput;	vp/vcXConnectIndexIn/ OutThroughput;
vp/vcCrossConnetRow- Status;		pathr/chanrStatus	pvcRowStatus;	vp/vcXConnectRowSta- tus
vp/vcCrossConnetAd- minStatus;				
			pvcCause	

Table 3.3 PVC MIBs and Common View

Chapter 4

Different Approaches to PVC Configuration Management

As discussed in the previous chapter, configuration management is one of the network management functional areas. PVC configuration management plays a very important role in ATM network management. Research activities in this area vary from integrating existing SNMP based upon configuration solutions to developing more innovative solutions such as using CORBA, Telecommunication Network Management (TMN) and JAVA.

This chapter analyses different approaches to PVC configuration management in heterogeneous ATM networks.

Section 4.1 Manager to Manager Based PVC Configuration Management

The manager and agent model is mentioned in chapter 2. This section analyses one approach to PVC configuration management in a multi-vendor ATM environment [19], [20].

First, the concept of the network management domain (domain) is to bind a set of managed objects to a specific manager. There can be several agents in a domain, each of which accesses a set of managed objects, as shown in figure 4.1. In the case of PVC provisioning, a domain is specific to a vendor, for example, Newbridge, while the managed objects are PVC related MIBs, for example, PVC cross connections.

A manager can access a managed object through a related agent. A managed object

can be accessed by only one agent. Also, if a manager tries to access a managed object in a remote domain, it has to communicate and negotiate with a remote manager. This is shown in figure 4.1. For multi-domain operation, the multi-domain management protocol (MMP) has to be defined.

Figure 4.1 Conceptual Model of Multi-domain Management

There are some implementations based on this model. For example, research work conducted by Bell Sygma developed an integration tool for ATM management. The tool manages Newbridge and GDC ATM switches. The key functionality of this tool is end-to-end PVC configuration management. Figure 4.2 illustrates the basic ideas of this research. In this system, there are only two domains, namely Newbridge and GDC. The 40620 and NMS 300 are the managers for Newbridge and GDC switches respectively. The NMS 300 manager accesses its GDC PVC resource through SNMP Apex agents. The 40620 man-

ager accesses its Newbridge xConnection resource via 36150s, which are Newbridge proprietary protocol agents. In order to integrate two managers, a communication protocol between the two managers needs to be defined.

The advantages of this approach in a heterogeneous ATM network are:

- use of existing manager software applications;
- only require implementing integration software between managers;
- rapid development

Because this approach uses existing software developed by the device vendors, there is no great need to develop new software, except the manager to manager communication software. It takes a relatively small amount of effort to integrate existing systems.

Figure 4.2 Example: An ATM Network Management Tool

The disadvantages of this approach are:

- need different manager software from different vendors;
- operator has to know all MIBs from different vendors;
- integration becomes difficult when the number of vendors grows

Although this approach requires only the integration of existing management systems, it is required to have all the manager software applications available. This can be expensive to a network composed of many different vendor's switches. Also, the network operator requires knowledge of each MIB for all the different switches. This is a needless burden on the operator and can cause human errors. Perhaps the biggest drawback to this approach is when the number of vendor grows, the integration among the multiple vendors will inevitably be difficult, if not impossible.

It should be noted that there is another way to handle this kind of multi-manager system. Instead of using a manager-to-manager protocol, a top level manager acting as central manager can be used to communicate with domain managers. These domain managers are mid-level managers and are the vendor specific products. In this model, a PVC configuration management task is delegated to the mid-level managers by the top level manager. The delegation is normally conducted using scripts. Unlike manager/agents model, there are no standard protocols, such as SNMP or CMIP, for the communication between the top level manager and mid-level managers. At the top of Figure 4.2, the dotted box represents the central manager, while two dotted lines represent the communication between the central manager and the Newbridge manager, and the central manager and the GDC manager.

Section 4.2 CORBA Based PVC Configuration Management

The second approach to PVC configuration management is the use of Common Object Request Broker Architecture (CORBA) [21]. The Object Management Group (OMG) initiated the Object Management Architecture (OMA) in recognition of the challenge of the developing software applications and components that support and make efficient use of heterogeneous networked systems [22]. CORBA is one of the key components of the OMA.

Two key components of CORBA are OMG's Interface Definition Language (IDL) and the IDL compiler. The IDL specification defines Object Oriented interfaces containing methods and attributes. It also maps IDL onto multiple programming languages, such as Java, C/C++, Smalltalk and etc. The OMG IDL compiler generates procedure stubs on the client side and procedure skeletons on the server side.

Due to its heterogeneous capability, some research work has been done in applying CORBA in network management. Figure 4.3 shows a possible management architecture in a multi-vendor ATM network environment.

In the case of PVC configuration management, the IDL represents the PVC configuration common view, which is discussed in chapter 3. The CORBA client, or the end human user only needs to know the common view. The vendor specific switch MIBs are transparent to the client. Interface IDLs distributively reside on different proxy machines. A client can invoke a method: `getAvailableBandwidth()`. Each IDL translates the available bandwidth from the common view to its vendor specific MIB object. For instance, Interface

IDL1 translates the available bandwidth from the common view to Fore MIB Object, throughput.

Some advantages of this approach in a heterogeneous ATM network are as follows:

- deploying CORBA enabled Web browser;
- supporting legacy systems;
- coexistence with current manager platforms

Making use of CORBA follows the trend to develop Web enabled software applications. With the wide use of WWW, this feature is obviously attractive. Supporting legacy systems is perhaps even more important. That means that instead of getting rid of the existing systems, they can be evolved to CORBA, for example from SNMP. Another aspect of using CORBA is coexistence with current manager platforms, for instance, HP Openview.

**Figure 4.3 An Architecture for Management Systems
in Heterogeneous ATM networks**

Although the advantages of this approach are appealing, there are some disadvantages.

They are as follows:

- a “heavy weight” solution;
- features depend upon services;
- overkill for small applications like PVC configuration

Since CORBA is intended for large network systems, it primarily uses heavy weight protocols. This may cause network traffic problems. Although on top of CORBA’s Object Request Broker (ORB), services like naming service can be applied to fast develop fea-

tures such as “get/set a MIB object”, these feature are normally expensive. The most important disadvantage of this approach is that for small application like PVC configuration, it is an overkill to use CORBA, because most of its features are not necessary.

Section 4.3 Mobile Code Based PVC Configuration Management

Another way to approach PVC configuration management is to use Mobile Agents. This innovative approach is built upon the mobile code infrastructure and simulator. Both are introduced in chapter 2. As a practical application, mobile code based PVC provisioning tackles challenging issues like heterogeneity and interoperability. Figure 4.4 illustrates the architecture of this approach.

Figure 4.4 PVC Configuration Management Based upon Mobile Code

In this approach, the PVC configuration manager sends a PVC agent to the switch at

one end of the network connection. The agent performs the PVC configuration tasks like setting up a resource, or releasing a resource. Then it migrates to the next switch in the route where it performs similar duty. The agent traverses all the switches in the route until it finishes all configuration tasks. For more information on how this approach works, see Chapter 5.

Compared with the previous two approaches, the mobile code based approach has outstanding advantages. First, there is no manager to manager software integration. The configuration manager has a common view of PVC configuration for all different switches. The operator does not have to be aware of the underlying switch systems from different vendors. Although vendor-specific MIB information is stored on each switch system, a SwitchInterface implemented as a VMC is used to translate the common view to the vendor specific MIB information. The PVC agent automates the connectivity procedures without requiring a human operator's decision making. Secondly, the PVC agents are designed in such a way that they are specialized to perform only PVC related tasks. Therefore, compared to monolithic agents in conventional manager/agents model, they are much smaller, thus cause less traffic in the network. Lastly, since Java is chosen to implement the software application, the development time is relatively small due to Java's OO, portability, networking and other features.

Chapter 5

PVC Configuration Management with Mobile Code in Heterogeneous ATM Environment

In the previous chapter, several different approaches to PVC configuration management in heterogeneous ATM networks were discussed, including integrating existing SNMP management applications, a CORBA based solution and a mobile code (JAVA) based solution. From the analysis of the advantages and disadvantages of those approaches, the mobile code based solution is superior to the other solutions. In addition to the advantages mentioned in the previous chapter, the PVC configuration management application fits well in the overall mobile code network management project, which was originated in the Network Management Lab in the System and Computer Engineering Department of Carleton University.

This chapter covers developing a PVC configuration management application based upon mobile code in heterogeneous ATM networks. It first proposes the architecture for a new way of PVC provisioning using mobile code. It then analyses the application software requirements, followed by high level design and detailed design and implementation. It also demonstrates using some examples.

Section 5.1 Architecture of Mobile Agent Based PVC Provisioning

Before the ATM cells transfer from one end of ATM network to the other end, a virtual

connection needs to be set up. In this thesis, only the Permanent Virtual Connection is considered.

In most cases, two enterprises make network connections through some network carrier. Figure 5. 1 shows the concept of the connection. The top part illustrates the two end switches from two enterprises connected through a carrier's ATM cloud. From the network point of view, a carrier's ATM network can be considered as one switch, according to ATM Forum M4 [23]. The lower part of Figure 5.1 depicts this idea.

Prior to sending ATM traffic, the network operator at one end configures the PVC at its switch as shown in Figure 5.2. He or she then contacts the carrier's operator, either by phone or by e-mail. The carrier's operator configures its switch(es) based on the information given by the end switch operator. After that, he or she contacts the network operator at the other end. The operator at this end finally configures the switch at this end.

Figure 5.1 ATM PVC Path

With the mobile code approach, only one operator is required. The human operator enters the end-to-end PVC configuration requirements, such as port connections for neighbouring switches and bandwidth. The PVC configuration application, which serves as a network manager, sends a PVC mobile agent to the network to conduct the configuration task. Depending upon the functionality of the task, the agent can be either a PVC setup netlet, a PVC release netlet, or a PVC trace netlet. The agent carries user requirements, accomplishes the configuration task at the switch 1 end first, then it migrates to the carrier's switch. It does the configuration task at the carrier's switch, then migrates to the switch 2 end. The agent finishes the task at the end switch 2, thus completing the end-to-end configuration task. Figure 5.2 illustrates this approach. It should be noted that the procedure is sequential rather than parallel. This is because in order to configure the carrier's switch or the end switch, a netlet has to finish the configuration on the previous switch first so that it can carry VPI/VCI values from one switch's outgoing port to next the switch's incoming port.

In a non mobile agent approach, each network operator needs to have the knowledge of the switch, in terms of its MIB. The monolithic agent residing on each switch actually does the work. In a heterogeneous ATM environment, this could be tedious. On the contrary, in the mobile agent approach, there is only one agent travelling through the network and conducting the configuration task. The operator only needs to know the general representation of a switch, or the "Common View" discussed in Chapter 3. At each switch, an interface, called the SwitchInterface is implemented as a VMC, which translates the "Common view" to the vendor specific PVC configuration data. For details see Section

5.3.

Figure 5.2 Non Mobile Code vs. Mobile Code Way of Configuring a PVC

Due to the unavailability of a real ATM network for testing the design, a simulation of the network is considered. Figure 5.3 is an example of a multi-vendor ATM network. In this network there can be one or more switches for each of three listed vendors. Here each PVC MIB is unique to a vendor specific switch. Each vendor specific switch interface (VMC), for example ForeInterface, plays a role of the interface between a vendor specific switch daemon and a simulated vendor switch MIB.

Figure 5.3 A Simulated ATM Network

On each switch, there is a daemon (MCD described in Chapter 2) running to accommodate the PVC agent, or PVCNetlet, which is injected by the PVCConfigurationManager. The PVCNetlet has access to a switchInterface through the switchDaemon. The switch interface maps the “Common View” of PVC knowledge to the vendor specific PVC MIB. The SimulatedSwitch actually sets the PVC configuration data. Figure 5.4 shows the relationship between the components in the PVC configuration application.

The top part of Figure 5.4 shows the components in a simulated environment. The four components inside the square box in the middle (Components in a simulated Environment) simulate a vendor specific ATM switch. The ATMSimulatorController is used to access the simulated PVC MIB on each switch. These MIBs can be stored in a central database and retrieved on demand. Each vendor’s switch interface is a VMC (see Chapter 2) and can be stored in a Universal Resource Location (URL). When a switch’s hardware is updated, a new interface can be brought from the vendor’s URL.

Components in a Simulated Environment

Components in a Non Simulated Environment

Figure 5.4 Model of Mobile Agent Based PVC Configuration Management

The lower part (Components in a Non Simulated Environment) of Figure 5.4 shows how a PVC Mobile Agent accesses a real switch which does not run the JAVA virtual machine. In this case, those components inside the square box in the middle run on a separate host with the real switch. The SwitchInterface needs to implement some native technique, such as Telnet, to access the real switch MIB.

Section 5.2 Application Software Requirements

This section analyses the requirements from the user's perspective for the PVC configuration management application software based upon mobile code. Since the purpose of the

thesis is to prove the concept of applying mobile code in PVC provisioning management, some of the detailed ATM aspects are taken into less consideration. The analysis makes some assumptions and is not exhaustive.

- **User interface**

A Graphical User Interface (GUI), PVCConfigurationManager is required for interacting between the human operator and the application. The GUI should be implemented as a JAVA Applet. It accepts the bandwidth and the ports information along the ATM PVC path. There is no routing decision to make. It is assumed that the operator has the knowledge of the starting switch, the transit switch and the end switch as shown in Figure 5.2. It is also assumed that only bandwidth is taken into consideration among all the QOS parameters when an end-to-end PVC needs to be setup. In other words, the type of the ATM traffic is going to be “best-effort”, as mentioned in Chapter 2.

- **Initial network setup**

When the network is created, all ports on each switch are initialized. The PVCConfigurationManager does not dynamically discover the network. The human operator must have knowledge of port inter-connections between switches in the network.

- **Setup end-to-end PVC**

The operator should be able to just specify the ports on start, transit (carrier) and end switch and required bandwidth. The end-to-end PVC setup is completed in two stages. Stage one involves the resource reservation, while stage two involves the resource commitment. The software should first check the resource (bandwidth, VPI/VCI) availability on each switch. If there are insufficient resources, an error message should be displayed. After that, the software should allocate the resources along the path

including the start switch, the end switch and the transit switch.

- **Trace end-to-end PVC(s)**

The operator should be able to trace either certain end-to-end PVC or all end-to-end PVCs by specifying the ports on the start, transit and end switches. The software should display bandwidth, VPI and VCI values on each switch with specified ports on start, transit and end switches.

- **Release end-to-end PVC(s)**

The operator should be able to remove a designated, previously allocated end-to-end PVC or all end-to-end PVCs by specifying the ports on start, transit and end switches. The software should remove those PVC(s), and release the previously allocated resources on each switch with specified ports on start, transit and end switches.

Section 5.3 Application Software Analysis

This section analyses the requirements from the software designer's perspective for the PVC configuration management application software.

Section 5.3.1 Mapping PVC Common View to Vendor Specific MIB

In the requirements analysis from the user's or network operator's perspective, the end user interface is transparent in terms of the underlying vendor specific switch. All the end user is aware of is the general switch PVC representation, or the "Common View" (which is analysed in Chapter 3). Because the actual PVC configuration needs to be done at each vendor specific switch, it is required to have a kind of mapping, or translation from the "Common View" to a vendor specific switch PVC MIB.

To materialize this mapping, a software component, called `SwitchInterface`, is developed based upon the mobile code infrastructure. It is a subclass of abstract class `VirtualManagedComponent` and acts as a bridge between a mobile PVC Configuration agent, `PVCNetlet`, and a vendor specific switch MIB. It is preinstalled with a switch netlet daemon. Therefore, when the `PVCConfigurationManager` wants to conduct some task, say setting up an end-to-end PVC, a `PVCNetlet` with a “Common view” is launched by the application manager to a switch daemon at the start end of the network path. The netlet then gains access to the `SwitchInterface`. The `SwitchInterface` in turn obtains access to the underlying switch. After completing the task on one switch, the netlet migrates to the next switch. Figure 5.5 depicts the procedure.

Figure 5.5 Mapping Common View to Vendor Specific Switch

For more details on the `SwitchInterface` component, see Section 5.4 and Section 5.5.

Section 5.3.2 Scenario Analysis

Based upon the user's requirements from the last section, several end-to-end PVC configuration scenarios, namely PVC set up, PVC trace and PVC release, are analysed in terms of preconditions, postconditions, assumptions and algorithms:

- **Set up end-to-end PVC**

In this scenario, the end user (network operator) enters the port information for the start switch, the transit switch and the end switch, plus the required bandwidth for the traffic. The PVC setup is conducted at the allocate resource stage and the commit resource stage.

Assumption:

The traffic class is UBR, in other words, it is "best-effort" traffic. Among all the QOS parameters, only the maximum bandwidth needs to be assigned prior to sending actual traffic at the virtual connection stage.

Preconditions:

- The physical port connections are known;
- Each port's resource capacity, i.e. the number of VPI/VCI and maximum bandwidth (available bandwidth) are given.

Postconditions:

- Each switch's cross connection is set up if there is resource available. Otherwise, exception message is displayed.

Algorithm:

allocateResource()

at the start switch, find the outgoing port

at the port, search for the first available VP with the required bandwidth

at the VP, search for the first available VC with the required bandwidth

return VPI and VCI;

at the transit switch

find the incoming port

at the port, search for the first available VP with the required bandwidth

at the VP, search for the first available VC with the required bandwidth

find the outgoing port

at the port, search for the first available VP with the required bandwidth

at the VP, search for the first available VC with the required bandwidth

return incoming VPI and VCI, and outgoing VPI and VCI;

at the end switch, find the incoming port

at the port, search for the first available VP with the required bandwidth

at the VP, search for the first available VC with the required bandwidth

return VPI and VCI;

commitResource()

for each switch, do

crossConnection

addConnection

decreaseResource

- **Trace end-to-end PVC(s)**

In this scenario, the end user (network operator) enters the port information for the start switch, the transit switch and the end switch.

Preconditions:

- There exists PVC(s) at the given start switch and end switch via the transit switch.

Postconditions:

- A collection of PVCs at given path are retrieved.

Algorithm:

at the start switch, find the outgoing port

at the port, find VP

at the VP, find VC

return the port number, VPI, VCI and bandwidth at the outgoing port;

at the transit switch

find the incoming port

at the port, find VP

at the VP, find VC

find the outgoing port

at the port, find VP

at the VP, find VC

return the port number, VPI, VCI and bandwidth at the incoming and the outgoing

port;

at the end switch, find the incoming port

at the port, find VP

at the VP, find VC

return the port number, VPI, VCI and bandwidth at the incoming port;

- **Release end-to-end PVC(s)**

In this scenario, the network operator wants to free the PVC between the start switch and the end switch, which has been previously set up. The consequence of this scenario is: along the PVC path, all related resources, i.e bandwidth, VP and VC at the each switch are released.

Assumption:

There is no traffic along the path.

Preconditions:

- There exists PVC(s) at the given start switch and end switch via the transit switch.

Postconditions:

- The end-to-end PVC is removed and all resources are reclaimed.

Algorithm:

at each switch do

find the specified cross connection

remove the cross connection

increaseResource

Section 5.4 High Level Design

This section covers the high level design issues of the software implementation. It describes the required software components in the mobile code based PVC configuration application and the necessary classes.

Section 5.4.1 Software Components

Figure 5.4 illustrates all the software components in the PVC Configuration Management Application. This thesis develops the application in a simulated environment. Therefore the access to a real vendor switch is not covered. From the figure, software components `SimulatedSwitch`, `SwitchInterface`, `PVCNetlet`, `SwitchDaemon`, `PVCConfigManager` and `ATMSimulatorController` are the core components.

The `SwitchDaemon` is implemented as a MCD, which is a part of the mobile code infrastructure. To extend the `SwitchDaemon` from the general MCD, the only consideration is the configuration file, which contains important parameters, such as the TCP/UDP listening port, `SwitchInterface` VMC, `SimulatedSwitch` VMC and migration route, in order to accommodate the `PVCNetlet`.

The following discusses the remaining core components of the PVC configuration management application based upon mobile code in a simulated environment.

SimulatedSwitch:

This component is a subclass of `VirtualManagedComponent`, which contains the vendor specific PVC MIB. It is decomposed into ATM physical layer interfaces, which in turn contain multiple virtual path links. Furthermore a virtual path link can be decom-

posed into multiple virtual channel links. Both virtual link and virtual channel link are at the ATM layer. Depending upon the vendor, a physical layer interface can be a port, or a line card. For each vendor, a more specific simulated switch is required. For instance, a ForeSimulatedSwitch, which contains multiple physical ports, is required to simulate a switch by Fore.

SwitchInterface:

This component is also a subclass of VirtualManagedComponent which translates the general representation, or the “Common View” to a vendor specific PVC MIB. It has access to the SimulatedSwitch. For each vendor, a more specific switch interface is required. For instance, a ForeSwitchInterface, which contains an access to a ForeSimulatedSwitch, is required to map the Common View to Fore specific switch in a simulated manner. This VMC can be expanded to perform other tasks like updating switch software

PVCNetlet:

This component is a subclass of Netlet, which performs PVC configuration related tasks. A PVCNetlet only has knowledge of the Common View of a switch. It is not aware of the underlying vendor specific switch. When a PVCNetlet is injected by a PVCConfigManager it carries user requirements for a configuration task. After the Netlet arrives on a SwitchDaemon, it obtains the reference pointer of the SwitchInterface. It then passes the user requirements to the SwitchInterface. The latter in turn passes the requirement to the SimulatedSwitch. The SimulatedSwitch finally configures itself. PVC configuration task can be setting up connection, trace connec-

tion or release connection. Depending upon the configuration task, a specialized PVCNetlet can be used. Three kinds of these Netlets, namely PVCSetupNetlet, PVCTraceNetlet and PVCReleaseNetlet, are developed to achieve the three different goals. A PVCNetlet can migrate from one switch to another depending upon the configuration task. For more detail, see Section 5.5.

PVConfigManager:

This graphical component serves as an interface between the end user, or network operator, and the application software. It takes end-to-end PVC configuration requirements from the user (such as port number and the bandwidth) and launches a specialized PVCNetlet to a start switch to perform one of three tasks: setting up a PVC, tracing PVCs or releasing a PVC. In addition, the PVConfigManager needs to know the IP address of a start switch.

ATMSimulatorController:

This component is a modified version of SimulatedControlProgram (SCP), which is mentioned in Chapter 2. The purpose of this component is to have access to the individual vendor specific PVC MIB. This graphical component can display each simulated switch's PVC MIB. In addition, as an alternative means of installing SimulatedSwitch VMC and SwitchInterface VMC on each SwitchDaemon, it can ship these two VMCs to each SwitchDaemon, which does not install the two VMCs at its starting up stage.

Section 5.4.2 Class Diagrams

In the previous subsection, the functionality of each core component in PVC Configuration Management application was discussed. This subsection analyzes these components from an Object-Oriented view. All the class diagrams use Object Modeling Technique (OMT) notation [24]. Rectangular boxes represents objects. The association among objects is represented by a line between them. The diamond at the end of a line represents a containment relationship. The diamond is located next to the object which is contained in the other.

Figure 5.6 is the class containment diagram for the Common View of an ATM switch. Each ATMSwitch object contains multiple Port objects and multiple CrossConnection objects. A Port can be at either the incoming side or outgoing side of an ATMSwitch. Each Port object is made of multiple VPLink objects, which in turn consist of multiple VCLinks. A CrossConnection object has fields: index, inPortNum, inVpi, inVci, outPortNum, outVpi, outVci and bandwidth.

Figure 5.6 Common View of ATM Switch

Figure 5.7 is the class inheritance diagram for simulated switches. The SimulatedSwitch object is a subclass of VirtualManagedComponent object, while the SimulatedForeSwitch, SimulatedCiscoSwitch and SimulatedNewbridgeSwitch are subclasses of SimulatedSwitch.

Figure 5.7 Simulated Switches

Similar to Figure 5.7, Figure 5.8 is the class inheritance diagram for switch interfaces. The SwitchInterface object is a subclass of VirtualManagedComponent, while the ForeSwitchInterface, CiscoSwitchInterface and NewbridgeSwitchInterface are subclasses of SwitchInterface.

Figure 5.8 Switch Interfaces

Figure 5.9 shows the class inheritance diagram for PVC netlets. The PVCNetlet object is subclass of Netlet, while PVCSetupNetlet, PVCTraceNetlet and PVCReleaseNetlet are

subclasses of PVCNetlet.

Figure 5.9 PVC Netlets

Figure 5.10 illustrates the class relationship of the PVC Configuration Management Application. A PVCConfigManager contains either PVCSetupNetlet, PVCTraceNetlet or PVCReleaseNetlet. Depending upon the configuration task, one of three PVCNetlet has access to ForeSwitchInterface, CiscoSwitchInterface and NewbridgeSwitchInterface. These three switchInterfaces contains SimulatedForeSwitch, SimulatedCiscoSwitch and SimulatedNewbridgeSwitch respectively.

Figure 5.10 Class Relationship of PVC Configuration Management Application

Section 5.5 Detailed Design

This section covers the design issues in a more detailed manner . The interaction of different classes in a PVC configuration management application is illustrated in message interaction diagrams for three different scenarios, namely setting up a PVC, tracing PVCs and releasing a PVC. The high level algorithms for these scenarios are shown in Section 5.3.

Section 5.5.1 Setting Up End-to-end PVC Scenario

In this scenario, objects `PVCConfigManager`, `PVCSetupNetlet`, `ForeSwitchInterface`, `SimulatedForeSwitch`, `CiscoSwitchInterface`, `SimulatedCiscoSwitch`, `NewbridgeSwitchInterface` and `SimulatedNewbridgeSwitch` participate in an end-to-end PVC set up. Figure 5.11 illustrates the message exchanges among these objects.

The end-to-end PVC setup is accomplished in two stages. In the first stage, the necessary resources i.e. bandwidth, VPI/VCI, are reserved at each vendor specific switch. Then in the second stage, the reserved resources are allocated. The advantage of dividing this procedure into two phases is that in case there are insufficient resources at a certain switch, there is no need to deallocate the resources at a previous switch. For example, the resources are allocated at the `SimulatedForeSwitch` first, but there are not enough resources at `SimulatedCiscoSwitch`. In order to undo the allocation of the resources at the `SimulatedForeSwitch`, the `PVCSetupNetlet` needs to go back that switch to release the resources.

Section 5.5.2 Tracing End-to-end PVC Scenario

In this scenario, objects PVCCConfigManager, PVCTraceNetlet, ForeSwitchInterface, SimulatedForeSwitch, CiscoSwitchInterface, SimulatedCiscoSwitch, NewbridgeSwitchInterface and SimulatedNewbridgeSwitch participate in end-to-end PVC tracing. Figure 5.12 illustrates the message exchanges among these objects.

Section 5.5.3 Releasing End-to-end PVC Scenario

In this scenario, objects PVCCConfigManager, PVCReleaseNetlet, ForeSwitchInterface, SimulatedForeSwitch, CiscoSwitchInterface, SimulatedCiscoSwitch, NewbridgeSwitchInterface and SimulatedNewbridgeSwitch participate an end-to-end PVC releasing. Figure 5.13 illustrates the message exchanges among these objects.

Figure 5.11 Message Interaction Diagram for Setting up PVC Scenario

Figure 5.12 Message Interaction Diagram for Tracing PVC Scenario

Figure 5.13 Message Interaction Diagram for Releasing PVC Scenario

Section 5.6 Implementation

The prototype implementation for mobile code based PVC configuration management application is written in JAVA. For detailed source code, see Appendix.

Section 5.7 Testing Examples

This section illustrates some testing examples of the PVC configuration management application software, including **setting up end-to-end PVC** and **tracing end-to-end PVC(s)**. The program simulates a Fore Runner switch at one end and a Newbridge 36150 switch at the other end with a Cisco LightStream switch as a carrier's switch (transit switch). Figure 5.14 is the image of the user interface of the PVC configuration manager.

The testing is run on a Pentium PC with Window NT. To run the program, the proper class paths, where all required JAVA classes are, need to be set up first. In order to run three switch netlet daemons, which simulate three switches, three DOS consoles are required. Change to the directory where SwitchDaemon configuration files are located and type the following at these consoles:

At DOS Console1 (simulates a Fore's Runner switch):

[type]: java mct.NetletDaemon startPVCnetletdaemon.properties

You will see the daemon installation messages like:

mct.admin.MobileCodeManager: Instantiator is installed

pvc.vmc.mib.SimulatedForeSwitch has been installed successfully

pvc.vmc.switchInterface.ForeSwitchInterface has been installed successfully

mct.admin.Listener: Waiting for TCP connection at port 8998

Applet Viewer: PVConfigManager.class

Applet

Vendor Switches:	Connection Parameters:
Start Switch: Fore Runner	Start Outgoing Port <input type="text" value="2"/>
Carrier's Switch: Cisco LightStream	Carrier's Incoming Port <input type="text" value="1"/>
End Switch: Newbridge 36150	Carrier's Outgoing Port <input type="text" value="3"/>
	End Incoming Port <input type="text" value="2"/>
	Bandwidth (in BPS) <input type="text" value="100"/>

Start Switch TCP/IP:

Host TCP Port

End-to-end PVC Configuration Functions:

Applet started.

Figure 5.14 PVConfigManager

At DOS Console2 (simulates a Cisco's LightStream switch):

```
[type]: java mct.NetletDaemon transitPVCnetletdaemon.properties
```

You will see the daemon installation messages like:

```
mct.admin.MobileCodeManager: Instantiator is installed
```

```
pvc.vmc.mib.SimulatedCiscoSwitch has been installed successfully
```

```
pvc.vmc.switchInterface.CiscoSwitchInterface has been installed successfully
```

```
mct.admin.Listener: Waiting for TCP connection at port 7998
```

At DOS Console3 (simulates a Newbridge's 36150 switch):

```
[type]: java mct.NetletDaemon endPVCnetletdaemon.properties
```

You will see the daemon installation messages like:

mct.admin.MobileCodeManager: Instantiator is installed

pvc.vmc.mib.SimulatedNewbridgeSwitch has been installed successfully

pvc.vmc.switchInterface.NewbridgeSwitchInterface has been installed successfully

mct.admin.Listener: Waiting for TCP connection at port 6998

Now you have three simulated switches running and are ready to launch PVCNetlets to perform PVC configuration tasks.

To set up an end-to-end PVC at the above simulated ATM network connection, click the **Setup** button at the PVConfigManager. You will see the following results at the three DOS consoles:

At DOS Console1 (simulates a Fore's Runner switch):

PVCSetupNetlet got the handle of pvc.vmc.switchInterface.ForeInterface

Start switch at outport: 2

Bandwidth: 100

CrossConnection: -1 -1 -1 2 0 0 100

About to migrate...

Note: At CrossConnection output, there are seven fields. The first three fields represent the port number, VPI and VCI at the incoming port. The next three fields represent the port number, VPI and VCI at the outgoing port. The last field represents the bandwidth for this PVC. Here the first three fields are -1, which means the incoming port information is not applicable.

At DOS Console2 (simulates a Cisco's LightStream switch):

PVCSetupNetlet got the handle of pvc.vmc.switchInterface.CiscoInterface

Transit switch at inport: 1

Transit switch at outport: 3

Bandwidth: 100

CrossConnection: 1 0 0 3 1 0 100

About to migrate...

At DOS Console3 (simulates a Newbridge's 36150 switch):

PVCSetupNetlet got the handle of pvc.vmc.switchInterface.NewbrdgeInterface

End switch at inport: 2

Bandwidth: 100

CrossConnection: 2 1 0 -1 -1 -1 100

To trace end-to-end PVC(s) at the above simulated ATM network connection, click the **Trace** button at the PVCConfigManager. You will see the following results at the three DOS consoles:

At DOS Console1 (simulates a Fore's Runner switch):

PVCSetupNetlet got the handle of pvc.vmc.switchInterface.ForeInterface

Start switch at outport: 2

CrossConnection: -1 -1 -1 2 0 0 100

About to migrate...

At DOS Console2 (simulates a Cisco's LightStream switch):

PVCSetupNetlet got the handle of pvc.vmc.switchInterface.CiscoInterface

Transit switch at inport: 1

Transit switch at outport: 3

CrossConnection: 1 0 0 3 1 0 100

About to migrate...

At DOS Console3 (simulates a Newbridge's 36150 switch):

PVCSetupNetlet got the handle of pvc.vmc.switchInterface.NewbridgeInterface

End switch at inport: 2

CrossConnection: 2 1 0 -1 -1 -1 100

The examples presented here show the results under normal circumstances. In the example showing the setup of a PVC, there are sufficient resources at each ATM switch for the end-to-end connection to succeed. If there are insufficient resources at some switch, an error message will be displayed.

In the example showing tracing a PVC, the cross connection information is displayed at each switch because the connection was set up prior to the request for a cross connection trace. If the request came before the setting up of the cross connection, an error message at each switch will indicate that there is no cross connection.

Chapter 6

Conclusions

This chapter concludes the thesis. It first summarizes the thesis' work in terms of the contributions to the current PVC configuration management. It then discusses some future work.

Section 6.1 Summary

Network configuration management is a difficult task. It becomes even more complex in heterogeneous ATM networks. In this thesis, a new approach to PVC configuration management is proposed. The new approach is an application to the Mobile Code infrastructure developed in the Network Management & Artificial Intelligence Lab of the Department of Systems and Computer Engineering.

In order to overcome requiring knowledge of the PVC configuration MIB for each individual switch in an ATM network for network operators, the thesis introduces a "Common view". With this PVC configuration "Common View", a network operator does not have to be aware of PVC configuration MIBs for different vendor switches. A piece of software component acts as an interface to map the "Common View" to each vendor specific PVC configuration data.

A software application, called PVC Configuration Management In Heterogeneous ATM Environments was developed. It was tested using Mobile Agents to perform end-to-end PVC configuration tasks, such as **setting up, tracing or releasing** end-to-end PVCs.

A prototype implementation of the software application was created. It consists of

twenty two classes and 6000 lines of JAVA code. With its graphical user interface, which is what a network operator would see, a user can choose among different PVC configuration functions. The underlying software launches a specific Mobile Agent depending upon the user's choice of PVC configuration tasks.

I believe the implementation of management features using Mobile Agent approach reduces the complexity of the implementation. The creation of a single purpose agent reduces the interactions among network management features, which can only reduce implementation time. This single purpose agent, because it is small, is also likely to be smaller than the data transferred among nodes as compared to the monolithic agent approach.

A drawback of the approach presented in this thesis is the inability to use it on real systems right now. The ability to migrate from one node to another in the network using a Mobile Agent is large achieved because the Java Virtual Machine (JVM) provides the device independence from the perspective of a computing platform on which the Mobile Agent is run. At this time, I am unaware of any JVMs for in-service ATM switches.

Section 6.2 Future Work

The scope of this thesis and the time constraints were such that it was impossible to implement every idea. The nature of the thesis work was experimental, which leads to iterative refinement of the software. It is hoped that the work will be carried on in the future, with the possible implementation of the following:

The current program is based upon simulating a multi-vendor switch ATM environ-

ment. The software component, which acts as an interface between the “Common View” of PVC configuration data and actual vendor switch PVC MIB does not have the capability to access PVC MIBs for real switches in a functioning ATM network. The software needs to be refined so to accommodate real ATM switches.

The ports connection between neighbouring ATM switches are static. An end user requires the port connection knowledge due to lack of auto discovery capability in the Mobile Code framework. After the auto discovery functionality is implemented, the user interface in the PVC configuration software should be able to launch a port connection Mobile Agent and display the connection data on the user interface.

Among all the ATM PVC connection parameters, only the maximum bandwidth is currently considered. In the future the program should be refined so that the application can incorporate other connection requirements, such as the Cell Error Ratio or Cell Loss Ratio for the ATM traffic.

References:

- [1]. *Integrated Broadband Networks -- An Introduction to ATM-based Networks*, Addison-Weisley, 1991
 - [2]. *Multimedia: Computing, Communications and Applications*, Innovative Technology Series, 1995
 - [3]. Uyles Black, *ATM: Foundation For Broadband Networks*, Prentice Hall Series, 1995
 - [4]. Martin DePrycker, *Asynchronous Transfer Mode: Solution For Broadband ISDN*, Ellis Horwood, 1995
 - [5]. The ATM Forum Technical Committee, *ATM User-Network Interface Specification V 3.0*, The ATM Forum 1993
 - [6]. Alexander, P. & Carpenter, k., *ATM Network Management: A status Report*, Data Communications pp.110-116, September 1995
 - [7]. Stallings, W., *SNMP, SNMPv2, and CMIP, The Pratical Guide to Network Management*, Addison-Wesley, 1993
 - [8]. Onvura, O., *Asynchtonous Transfer Mode Networks, Performance Issues*, Artech House, 1995
 - [9]. *Perpetuum Mobile Procura Project*, <http://www.sce.carleton.ca/netmanage/perpetuum.shtml>, 1997
 - [10]. Bieszczad, *Towards Plug-and play Networks with Mobile Code*, Tech. Report SCE-97-08, Dept. Systems and Computer Engineering, Carleton University, March 1997
-

- [11]. *The Java Virtual Machine Specification*, Tech. Report, Sun Microsystems, 1996
 - [12]. G Susilo, A Bieszczad, & B Pagurek, *Infrastructure for Advanced Network Management Based on Mobile Code*, 1996
 - [13]. Campeau A., *Managing Networks with Mobile Code*, Tech. Report SCE-97-09, Dept. Systems and Computer Engineering, Carleton University, April 1997
 - [14]. The ATM Forum, *ATM User-Network Interface Specification V3.1*, Prentice Hall, 1997
 - [15]. The ATM Forum Technical Committee, *Integrate Layer Management Interface V4.0*, The ATM Forum, 1996
 - [16]. M. Ahmed & K. Tesink, *Request for Comments: 1695*, Internet Engineering Task Force 1994
 - [17]. Fore Systems Inc., *forethought_3.4.x.mib*, (private communication) 1996
 - [18]. Cisco Systems Inc., *Cisco LightStream MIB*, (private communication) 1995
 - [19]. Kwang-Hui Lee, *Design and Implementation of a Configuration Management System*, Globe Telecom, 1993
 - [20]. Bell Sygma, *CANARIE ATM Network Management Integration Tools User Guide*, (private communication) April 1997
 - [21]. *The Common Object Request Broker: Architecture and Specification 2.0 ed.*, Object Management Group, July 1995
 - [22]. Steve Vinoski, *CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*, IEEE Communications Magazine, Vol.14, No.2, February, 1997
-

- [23]. ATM Forum Technical Committee, *M4 Interface Requirements and Logical MIB*, ATM Forum, October 1994
 - [24]. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy & W. Lorensen, *Object-Oriented Modeling and Design*, Pretice Hall, 1991
 - [25]. *Mobile Agent Facility Specification*, OMG TC Document cf/xx-x-xx, June 1997
 - [26]. Busse, I. Covaci, S., *Customer Facing Components for Network Management Systems*, Integrated Network Management V, Proceedings Fifth ISINM, San Diego, May 1997
-

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