MIBlets: A Practical Approach to Virtual Network Management

W. F. Ng, D. S. Jun, H. K. Chow, R. Boutaba and A. Leon-Garcia
Department of Electrical and Computer Engineering, University of Toronto
Toronto, Ontario, Canada, M5S 3G4
Phone: (416) 978-1611 Fax: (416) 978-4425
Email: {walfrey, ajun, keith, rboutaba, alg}@nal.utoronto.ca

Abstract

This paper introduces the MIBlet concept as a means for effectively designing and managing Virtual Networks (VNs). MIBlets are logical structures providing abstract and selective views of the physical network resources allocated to VN customers. They result from the partitioning of the network resources (their MIB representations) and restrict customers' access only to those resources allocated to them. Different partitioning schemes are supported to provide virtual network services with different quality of service requirements. Customer control/management functions are implemented through MIBlet controllers located at every network node involved in the customer network. MIBlet controllers enforce customers' access control and resource usage policing strategies and are invoked to set up, monitor and control the customer connections. In this paper the partitioning of network resources into MIBlets is mainly addressed in the scope of the ATM testbed as a part of the Network Resources Management (NRM) Project of the Network Architecture Laboratory at University of Toronto.

Keywords

Network programmability, virtual networks**,** resource partitioning, MIB

1. Introduction

The purpose of a Virtual Network (VN) is to allow a "*soft networking environment*". Soft networking environment means flexible resource capacity, configurable network topology, and programmable network management/control architecture. Ultimately, VNs are created for the use of customers who, in turn, provide network services to end-users. Depending upon network management/control objectives, each customer may want to have full control over network management/control algorithms and mechanisms for FCAPS[1] management, routing, and signaling. For that purpose, the Virtual Network Resource Management (VNRM) Architecture has been designed in the Network Architecture Lab (NAL) at the University of Toronto. Programmability in network management/control functions is supported through the dynamic binding of the Virtual Network Resources (VNRs) to any

1 Fault, Configuration, Accounting, Performance and Security management

management/control system of customer's choice with proper algorithms, mechanisms, and interfaces.

The implementation of the dynamic system binding depends on the availability of management/control interfaces to resources (e.g., switches). The lack of open interfaces hampered the fast introduction of innovative network technologies and functions, and hence the network programmability. To stimulate faster development and support of networking applications, research and standardization efforts are currently dedicated to the specification of open switching/routing architectures, signaling protocols and binding interfaces. For the time being the functions of network resources are mostly implemented by rigidly built-in software, which accesses vendor-proprietary instrumentation to set up, monitor and control network connections. This has led to a redefinition of the role of network management through MIBs to incorporate features traditionally engineered into embedded software. For example ATM switches initially provided access to VC instrumentation via SNMP MIBs and RSVP routers are controlled by management software. Currently, virtual LAN features are supported by switched LANs through management interfaces using MIBs to set up, monitor and control virtual LANs. The reason for that is the complexity of development of signaling protocols and associated software. More importantly, the time to market for management software is much less compared to that of signaling software.

The work presented in this paper falls into this category that is to use MIBs to handle VN design, control and management. The primary motivation is to practically demonstrate the feasibility and applicability of the generic VNRM architecture by utilizing available network protocols, mechanisms, and products. Accordingly, the focus of this paper lies on partitioning of the network resources of the ATM testbed available in our laboratory using SNMP MIBs. Typically, both the signaling software and the SNMP MIBs access and manipulate the same switch instrumentation. However, the ATM switch MIB already incorporates a large number of accessible configuration variables that can be used to effectively partition the network.

The partitioning of network resources is provided through the MIBlet concept. A MIBlet is a logical structure providing the VN it is assigned to with *abstract* and *selective* views of the network resources. The abstract view hides the details of the resource interface that are not relevant to the VN. The selective view restricts visibility to the VN management system to those parts of the network resource allocated to the VN.

The paper is structured into five sections. Section 2, briefly summarizes the VN concept and the VNRM architecture. Section 3 introduces the MIBlet concept and focuses on the partitioning of network resources. The fourth section shows some applications of VN and MIBlets considered as part of the ATM and IP over ATM testbed. Section 5 compares the MIBlet approach with some related works. Finally, section 6 concludes the paper and points out future research directions.

Glossary

VN: Virtual Network
VNR: Virtual Network Resource
NRM: Network Resources Management
VNRMS: Virtual Network Resources Management System
CNRMS: Customer Network Resources Management System

2. Virtual Network Resource Management

In this section, we briefly introduce the VN concepts and the Virtual Network Resources Management (VNRM) architecture, which has been developed at the University of Toronto (more details can be found in [1]). A Virtual Network (VN) is defined as a collection of Virtual Network Resources (VNRs), where a VNR is defined as a logical subset of a Physical Network Resource (PNR). PNRs may include transmission bandwidth, buffer size, switch processing power, address space (VPIs and VCIs for ATM), scheduling mechanism. In order to allow logical operations on network resources, PNRs in the physical domain are given logical representation to be projected into the virtual domain (Figure 1). This is called *abstraction* of network resources. Through abstraction processes, PNRs become Root[2]-VNRs. For the management purposes, processing of information is organized into two layers: *network management layer* and *resource management layer*. The abstraction process in the network management layer is translated into a series of abstraction processes in the resource management layer, through which a group of PNRs becomes a group of corresponding Root-VNRs.

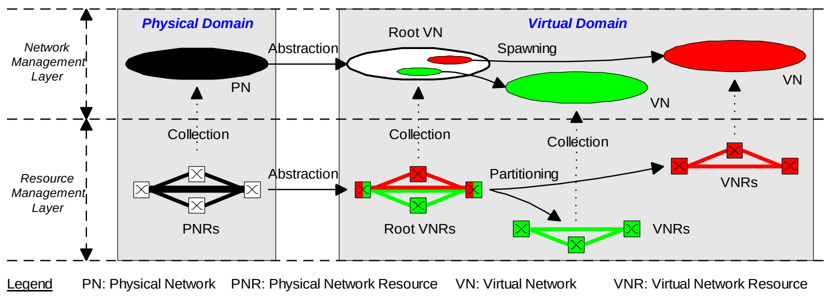

Figure 1: Physical and Virtual Networks and their Resources.

Figure 1 illustrates the relationships amongst a PN, VNs, PNRs, and VNRs. Through an abstraction process, a PN becomes a Root-VN. Once the Root-VN is established, multiple child VNs can be generated from the Root-VN through *spawning processes*. Spawning a VN corresponds to partitioning a group of VNRs: the notion of VNRs effectively translates the problem of creating a VN into the problem of creating a group of VNRs. It is generally assumed that aggregated capacity of child VNRs are less than or equal to the capacity of the parent VNR. At the expense of lower Grade-of-Service (GoS), however, oversubscription of resource capacity may be allowed to VN customers in order to leverage sharing of resources at the level of networks.

Ultimately, VNs are created for the use of customers who, in turn, provide network services to end-users. Depending upon network management/control objectives, each customer may want to have full control over network topology, resource capacity, and network management/control capability. The VNRM

[2] The term "Root" is used to emphasize that a Root-VNR is the very origin of other (child) VNRs.

architecture [1] is designed to support customization of these customer requirements at three different levels. At the level of resources, a Resource Agent flexibly assigns capacity to the network resource; and at the level of networks, the VNRM system flexibly configures the topology of a VN. Through the concept of *dynamic system binding*, another level of customization can be exercised for VNs. By selecting appropriate communication interfaces, a VN can be associated with a control system with desirable control algorithms and mechanisms.

With a Resource Agent, a VNR Controller controls a group of VNRs to interact with the corresponding Customer Network Resources Management System (CNRMS) for the provisioning of customer domain network services (Figure 2). By delivering abstract and aggregated information about status and connectivity of VNRs to the CNRMS, the Resource Agents effectively hide the non-essential details of what they represent. The operations of a VNR Controller voluntarily abide by rules and policies of its own Resource Agent, which is in the domain of the VN service provider. During the lifetime of the group of VNRs, the VNRs Controller performs usage control (or policing) to ensure conformance in the use of the VNRs to the capacity contract. Figure 2 shows one Root-VN and two child VNs. Each and every VNR Controller is directly accessed by the corresponding customer management/control system. Each VNR Controller serves its own CNRMS as an element of the virtual networking environment (VN) for the customer.

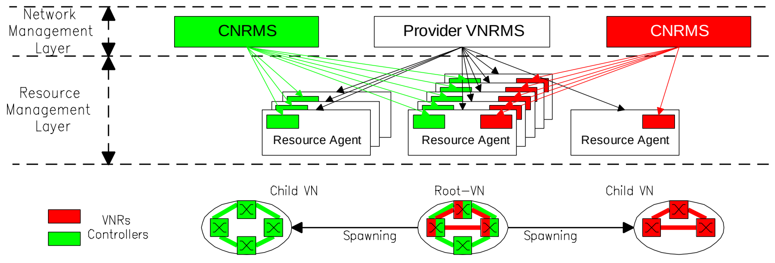

Figure 2: Dynamic System Binding.

The dynamic system binding technique is based on the concept of “open” switch control architecture, where a network switch provides a control interface with rich functionality and high flexibility. Thus, implementation of the dynamic system binding technique depends on the availability of management/control interfaces of switches. Since the significance of “open” control architectures was recognized only recently, the availability of network switches with “open” control interfaces has been very limited (if not nil). In fact, the widely available switch management/control interface is MIB (Management Information Base) through SNMP or CMIP. For this reason, we have chosen MIB and SNMP as pragmatic tools to realize the VNRM architecture.

3. MIBlet Concept

A network element contains a database that stores information about configuration, statistics and status of resources at the network element. This database is called a

Management Information Base (MIB). By manipulating values inside the MIB, we can monitor and control resources in the network element. In order to allow a CNRMS to manage only a subset of resources in a network element, the MIB of the network element is logically partitioned into multiple MIBs (*MIBlets*). The MIB of the network element can be effectively identified with Root Virtual Network Resources (Root-VNRs), and the MIBlet can be identified with Virtual Network Resources (VNRs) of a virtual network. The Resource Agent is responsible for providing *abstract* and *selective* views [2] to each CNRMS accessing it. An abstract view hides the details of the resource interface that are not relevant to the CNRMS. A selective view restricts the CNRMS to only access the resources allocated to it. To provide these views, instead of allowing a CNRMS to visualize all the managed objects in the MIB, the Resource Agent provides the CNRMS with only the managed objects that are relevant to the CNRMS. Also, the Resource Agent needs to control the access of the CNRMS's requests to manipulate the managed objects. This form of access control prevents the CNRMS from manipulating the managed objects in a way that violates its reservation condition. In Figure 3, the illustration shows that the MIB associated with the Root-VNR is logically partitioned into several MIBlets, each of which is dedicated to a CNRMS. The MIBlet concept is applicable to any network element. Since the testbed in the Network Architecture Lab (NAL) at the University of Toronto uses an ATM switch, we specify the details of MIBlet in terms of ATM.

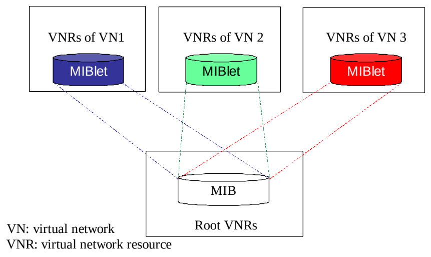

Figure 3: MIBlet Concept.

3.1. Resources Partitioning

The Partitioning of ATM switch resources is considered in this section. Resources considered for partitioning are: port, VPI/VCI space and bandwidth.

3.1.1. Port and VPI/VCI space Partitioning

Three levels of granularity have been proposed in [3]: port level, VPI level and VCI level and we adopt this scheme as part of our resources partitioning.

a) VCI level: each MIBlet is assigned certain VCI ranges on certain VPIs on certain ports.
b) VPI level: each MIBlet is assigned certain VPI ranges on certain ports.
c) Port level: each MIBlet can reserve certain ports within a switch.

Port level partitioning is easier to handle. However, the (virtual) network topology will be limited because each MIBlet is restricted to use only the assigned ports to make connections.

3.1.2. Bandwidth Partitioning

For bandwidth partitioning, two bandwidth types are defined: *Hard* bandwidth and *Soft* bandwidth.

a) Hard: Certain amount of bandwidth on each port is reserved for each MIBlet.

b) Soft: No bandwidth is reserved; bandwidth is allocated on a demand basis.

When a VNRMS requests a MIBlet creation, two values need to be specified on each port: hard bandwidth and soft bandwidth. Once the MIBlet creation request is accepted, a CNRMS can with certainty obtain the requested bandwidth if the bandwidth used by the CNRMS does not exceed the amount of hard bandwidth. When all the hard bandwidth is used up the CNRMS may request for an additional amount of bandwidth (soft bandwidth). For soft bandwidth, there is no guarantee that the requested bandwidth can be obtained. Section 3.3, which specifies the building blocks of the Resource Agent, will describe how bandwidth reservation can be achieved. Hard bandwidth reservation may result in inefficient bandwidth utilization, because bandwidth reserved by one MIBlet cannot be used by other MIBlets, even though the MIBlet that reserves the bandwidth is not using all the reserved bandwidth. However, this reservation scheme allows the MIBlet to guarantee that certain bandwidth can be assigned at the moment of request. To make the hard reservation scheme a bit more efficient, VNRMS may request for different amounts of bandwidth at different times, based on the timely profile of its network traffic.

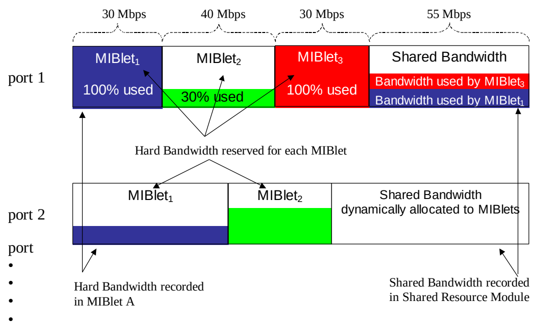
30 Mbps
40 Mbps
30 Mbps
55 Mbps
port 1
MIBlet$_1$
100% used
MIBlet$_2$
30% used
MIBlet$_3$
100% used
Shared Bandwidth
Bandwidth used by MIBlet$_3$
Bandwidth used by MIBlet$_1$
Hard Bandwidth reserved for each MIBlet
port 2
MIBlet$_1$
MIBlet$_2$
Shared Bandwidth dynamically allocated to MIBlets
port
Hard Bandwidth recorded in MIBlet A
Shared Bandwidth recorded in Shared Resource Module

Figure 4: Bandwidth Allocation.

The bandwidth related requests sent by the CNRMS are the requests to establish VPCs or VCCs with specified traffic contract parameters. Instead of directly sending the requests to the switch, the CNRMS sends the requests to its *MIBlet Controller*, which is responsible for controlling the access of the requests to the switch and is one of the building blocks in the Resource Agent. (The detail description of the MIBlet Controller will be presented in Section 3.3.3.) Figure 4 shows an example of bandwidth reservation. In this example, the total bandwidth for port 1 is 155Mbps. The hard bandwidth reserved by MIBlet_1, MIBlet_2 and MIBlet_3 C are 30Mbps, 40Mbps and 30Mbps, respectively. The remaining 55Mbps is considered a shared bandwidth and is dynamically allocated to the MIBlets. Every time the CNRMS sends a connection establishment request to its MIBlet Controller, the Controller will first check if there is enough hard bandwidth available. When hard bandwidth is not available, the MIBlet Controller will compete for shared bandwidth on behalf of the CNRMS. The example in Figure 4 shows that all the hard bandwidth reserved in MIBlet_1 is used up. When its CNRMS still wants to request for extra bandwidth, MIBlet_1 Controller will compete with other MIBlet Controllers for the 55Mbps shared bandwidth. An arbiter is required when MIBlet Controllers compete with each other for the shared bandwidth. Every time the arbiter receives requests from the MIBlet Controllers, it needs to check if there is enough shared bandwidth available. We call this arbiter the Resource Controller and the description of the Resource Controller will be presented in Section 3.3.2.

By computing the bandwidth usage, each MIBlet Controller in the Resource Agent can check if there is enough reserved hard bandwidth available for the connection establishment request and carries out connection admission control. Admission control here refers to the aggregate equivalent bandwidth computation approach. If the aggregate bandwidth usage reaches the reserved bandwidth, this means that the reserved bandwidth is used up. By using the same approach, the Resource Controller in the Resource Agent can check if there is enough shared bandwidth available. The admission control performed by the Resource Agent is preliminary and the final admission control is actually done by the switch. The responsibility of this preliminary admission control is to restrict each CNRMS to access only the bandwidth it reserves so that the CNMRS does not interfere with the bandwidth reserved by other CNRMSs. Moreover, it allows faster response time because the preliminary admission control can reject some of the requests that clearly violate the reservation condition. It is highly desirable for the Resource Agent to use the same admission control scheme as the switch. However, it is unlikely that the switch vendor would be willing to provide this sensitive information. Our ultimate goal is to integrate the Resource Agent into the switch. Since the connection admission control scheme of the switch is not available at this point, we use one of the equivalent bandwidth evaluation methods described in [4],[5],[6],[7],[8]. The equivalent bandwidth computation scheme does not have to be accurate or complicated because the Resource Agent is only responsible for performing a preliminary admission control. If the preliminary admission control is too optimistic, excessive bandwidth may be used by one CNRMS and the switch may reject requests from other CNRMSs later, even if the requests pass the preliminary admission control. If the preliminary admission control scheme is too pessimistic, requests may be blocked even if the resources are not fully utilized.

3.2. Proposed Architecture of Resource Agent

The Resource Agent design described below focuses on: (1) the creation of MIBlets, and (2) how MIBlet Controller carries out access control and provides the CNRMS with a limited view of the MIB. There are three functional building blocks identified in the Resource Agent: (1) Request Controller, (2) Resource Controller, and (3) MIBlet Controller. Figure 5 shows the architecture of the Resource Agent. Detailed description of the building blocks will be presented in Section 3.3. The interface to CNRMS can use a protocol such as SNMP or CMIP. The choice of the interface to a switch depends on the protocols supported by the switch.

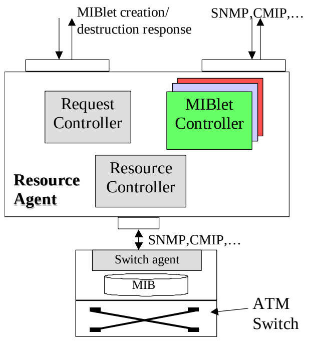

Figure 5: Architecture of Resource Agent.

3.2.1. Interfaces to Network Resource Management Systems

Two categories of interface specifications are defined: (a) requests for the creation, re-configuration, and termination of MIBlets, and (b) communication between a CNRMS and a MIBlet Controller.

(a) Requests for the Creation, Re-configuration and Termination of MIBlets

There are six messages related to the creation, re-configuration and termination of MIBlets: MIBlet creation request, MIBlet creation response, MIBlet re-configuration request, MIBlet re-configuration response, MIBlet termination request, and MIBlet termination response. The requests are sent from the VNRMS to the Resource Agent. For each of the requests, a corresponding response is returned from the Resource Agent to the VNRMS. The type definition for the parameters used in any of the six messages is stated first. The parameters required for each message will then be described.

```
Type Definition for the Parameters used in any of the Messages
enum      S_F          {success, failure};
struct    Port_Res     {
          int  Port;
          int  minVPI, maxVPI;
          int  minVCI, maxVCI;
          float     HardBandwidth;
          float     SoftBandwidth;
          };
typedef        enum    Control_Architecture  ControlArchitecture;
typedef        enum    S_F            SuccessIndication;
typedef        struct  Port_Res       PortReservation;
typedef        long                   ServiceAddress;
typedef        long                   RequestID;
```

MIBlet Creation Request: PortReservation parameters are specified for each port. Each PortReservation structure contains: (1) a port number that the VNRMS wants to reserve; (2) the VPI range that the VNRMS wants to reserve on the port specified;

(3) the VCI range that the VNRMS wants to reserve on the VPIs of the specified port; (4) the amount of hard bandwidth, and (5) the amount of soft bandwidth required on the specified port. Lastly, the Request ID is specified in the request. The Request ID is an integer used to match a response with its request. Since the response and its request have the same Request ID, the VNRMS can match the response received from the Resource Agent with its corresponding request using this Request ID.

MIBlet Creation Response: The Request ID is specified in the message so that the VNRMS can use this ID to match the response with its request. Also, the Success Indication is specified to indicate whether the request is accepted or not. If the request is accepted, a Service Address[3], which is used by the VNRMS to communicate with its MIBlet Controller, is included in the response.

MIBlet Re-configuration Request: A CNRMS may require a different amount of bandwidth and network topology at different times, based on the timely profile of its network traffic. The MIBlet re-configuration request allows the VNRMS to change the configuration during the lifetime of the MIBlet on behalf of the corresponding CNRMS. The parameters that need to be specified in the message are the same as those specified in the MIBlet creation request. In addition, the Service Address of the MIBlet Controller is specified in the message.

MIBlet Re-configuration Request: The Success Indication and the Request ID are specified in the response.

MIBlet Termination Request: The Service Address of the MIBlet that the VNRMS wants to terminate is specified in the request. Also, the Request ID is included.

MIBlet Termination Response: The Success Indication and the Request ID are specified in the response.

(b) Communication between CNRMS and MIBlet Controller

Since the testbed in our laboratory uses SNMP, we specify the details with the assumption that the protocol used to communicate between CNRMS and MIBlet Controller is SNMP. Thus, the messages used by a CNRMS to interact with MIBlet Controllers are standard SNMP messages, e.g. *get-request* and *set-request*. By manipulating parameters inside the MIBlet, the CNRMS can access the switch status and configure VPCs and VCCs. The CNRMS accesses the MIBlet as if it is directly accessing the MIB in the switch. However, the CNRMS can only access the values related to the reserved resources. The MIBlet Controller will provide the CNRMS with a MIB definition (called MIBlet definition), which defines the structure of the MIB objects. The MIBlet definition provided by the MIBlet Controller to the CNRMS is very similar to the switch's MIB definition provided by the switch vendor. The MIBlet definition must keep all the important MIB group in the switch's MIB definition such as: the *Port Group* which contains the managed objects concerning the ports, the *Path Group* which specifies the managed objects concerning the virtual paths, and the *Channel Group* which contains the managed objects concerning the virtual channels. Some less important MIB groups, such as

[3] Service Address here refers to socket port number which is used by the operating system to match the messages with the application

Temp Group which contains the temperature sensor information of the switch, may be omitted in the MIBlet definition.

3.3. Specifications of the Building Blocks

This section presents the specification and modeling of the three functional building blocks: (1) Request Controller, (2) Resource Controller, and (3) MIBlet Controller. The description of the building blocks is chosen to be SNMP specific. However, the conceptual model is generic. If a protocol other than SNMP is used later as an interface, the conceptual model will be similar and only the components involved in the protocol are modified. The specifications below are not intended to be exhaustive, but rather to capture the major features of the building blocks involved.

3.3.1. Request Controller

The Request Controller is invoked whenever a MIBlet creation request generated by the VNRMS is received. Upon receiving a MIBlet creation request, the Request Controller examines the request against pre-configured policies. An example of the pre-configured policy is to limit the maximum amount of bandwidth that can be reserved by any VNRMS. If the request does not comply with the policies, the Request Controller sends a MIBlet creation response to the VNRMS indicating that the request is rejected. If the request complies with the pre-configured policies, the Request Controller checks to see if there are enough resources to create the requested MIBlet. If there are not enough resources, the controller sends a MIBlet creation response to the VNRMS indicating that the request is rejected. If there are enough resources, (a) a service address, which is to be used by the CNRMS to communicate with its MIBlet Controller, is assigned to the newly created MIBlet Controller; (b) a MIBlet creation response, encapsulating the service address of the MIBlet Controller, is sent to the VNRMS to indicate that the request is accepted; (c) reservation parameters are fed into the newly created MIBlet Controller.

3.3.2. Resource Controller

The major responsibility of the Resource Controller is to act as an arbiter when MIBlet Controllers compete with each other for shared resources. When the MIBlet Controllers need to compete for shared resources on behalf of their CNRMSs, the MIBlet Controllers will send the set-requests to the Resource Controller. Section 3.3.3, which presents the functions of the MIBlet Controller, will describe clearly when the MIBlet Controller will send the request to the Resource Controller.

The set-requests received from different MIBlet Controllers are handled one by one. The Resource Controller contains the amount of the shared bandwidth for each port. Every time the controller receives a request for connection establishment, it computes the aggregate bandwidth usage for having this new connection. When the aggregate bandwidth usage reaches the amount of shared bandwidth, this implies that the shared bandwidth is used up and further requests will be rejected. If the request is accepted, (a) the set-request is sent to the switch SNMP agent; (b) the aggregate bandwidth is computed for accepting this new connection. If the request is rejected, the Resource Controller will send a response to the CNRMS, notifying that the request is rejected.

3.3.3. MIBlet Controller

Once the MIBlet creation request is accepted by the Request Controller, the CNRMS can send requests, such as VC segment establishment and enquiry for switch information, to its MIBlet Controller. The requests are handled one by one. The MIBlet Controller contains information about ports and VCI/VPI ranges reserved by its CNRMS. It also contains the amount of hard and soft bandwidth reserved on each port. For requests not related to bandwidth, the controller checks to see if the requests comply with the reservation condition. For bandwidth related requests, the controller checks to see if the reserved hard bandwidth is used up or not.

If hard bandwidth is available, the set-request is then sent to the switch SNMP agent. If hard bandwidth is used up while soft bandwidth is still available, the MIBlet Controller will send the request to the Resource Controller in order to compete for the shared bandwidth. If both hard and soft bandwidth are used up, the request is rejected and a response is sent to the CNRMS notifying that the request is not successful.

4. Applying VNs and MIBlets

This section briefly discusses how the VN and MIBlet concepts can be applied to real situations. The first subsection introduces our testbed network to demonstrate the applicability of the concepts, and the second subsection discusses migration of network control architectures and software infrastructures for increased programmability and flexibility in switch control.

4.1. Partitioning of NAL Testbed Network

We are currently building a testbed to construct three VNs over an ATM network. As illustrated in Figure 6, each VN is to be utilized for a distinct purpose. One is to serve an Integrated Services IP (IS-IP) network; another is to serve a Best Effort IP (BE-IP) network; and the other is to serve a native ATM network. As an independent project, an IS-IP control architecture (called ISAC [9]) has been recently developed and implemented in NAL at the University of Toronto, using RSVP, SNMP, and IS-IP/ATM service mapping. In order to demonstrate the customizability of the MIBlet-based VNRM architecture, we host the ISAC architecture on top of a VN instead of a physical ATM network. The BE-IP VN will also support ordinary data applications such as ftp, telnet, and www. The ATM VN will serve native ATM applications.

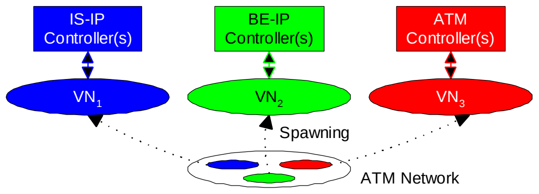

Figure 6: Partitioning of NAL Testbed.

In order to spawn three VNs of the NAL ATM network, three MIBlets are created from a MIB of each ATM switch with limited views and capacities, namely MIBlet$_1$,

$MIBlet_2$, and $MIBlet_3$ to support IS-IP, BE-IP, and ATM networks, respectively. In particular, VPI/VCI space and bandwidth of a MIB are partitioned and allocated to three MIBlets. After each switch's MIB is partitioned to MIBlets, a VN can be simply represented as a collection of MIBlets. VN_1 is the collection of all the $MIBlet_1$'s; VN_2 is a collection of all the $MIBlet_2$'s; and so forth. Once a VN is created with a subset of network resources, the VN can be utilized to serve its own purpose.

Ideally, it is desirable to have full customization of a VN to include the dynamic binding of a control system. In reality, however, customization of control system is limited by the interface to access the switches. Since ATM switch in NAL only supports SNMP connections for the purpose of management and ISAC uses SNMP as a control interface, ISAC has been chosen to provide IS-IP service. As far as the ISAC controller is concerned, the MIBlet is just a MIB that it can access. Therefore, no change is required for the ISAC controller. Since ISAC switch controllers employ RSVP as a signaling protocol and RSVP delivers control messages through BE-IP services, the BE-IP VN is required for the operation of ISAC controllers. For the time being, BE-IP and ATM control mechanisms and services provided by the NAL ATM switch are used for the control of VN_2 and VN_3.

4.2. Migration to Programmable Node

The ultimate goal of the VNRM architecture is to provide a fully programmable networking environment. Dynamic configurability in network topology and resource capability is enabled by the introduction of intelligent agents (Resource Agents), which perform delegated tasks of MIBlet management/control in an autonomous manner. This migration from an ordinary management/control architecture to an agent-based VN management/control architecture is depicted in Figure 10 (a) and (b). Note that there is no architectural change required for the IS-IP controller. Rather, the Resource Agent transparently fulfills partitioning of resources, arbitration of resource usage, and access control.

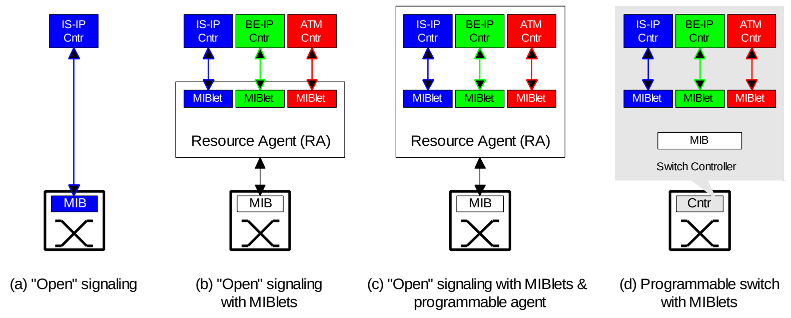

Figure 7: Node Migration from "Open" to "Programmable".

Compared to the traditional "closed" approach for switch control, the "open" approach has a number of advantages to include independent development of control algorithms and mechanisms from switch hardware, and dynamic binding of control architectures to switches. However, the physical separation of the control software from the switching hardware introduces a few extra communication procedures, which may impact the performance of overall control systems. One way to eliminate them is to employ programmability in management/control architectures (Figure 7 (c) and (d)). By defining appropriate service primitives and APIs in a node, local management/control functions can be easily implemented and deployed in the node. In order to fully automate code deployment within a node, a mobile code hosting environment can be included in the node[4]. In this way, new management/control functions and algorithms can be downloaded into a node from outside, emulating the "open" control environment without performance penalty. This programmable environment can be achieved by integrating a Java mobile code hosting capability [12] into the Resource Agent.

5. Related Work

Recently, proposals on developing a virtual network device model are reported. This model involves software entities that are (1) logical representations of certain state variables and (2) abstractions of resources of the underlying physical devices. Examples are the Binding Interface Base (BIB) [10] and the Switchlet [3]. The BIB is an organized collection of interfaces that model network resources. These interfaces provide resource abstractions for creating network services. [3], on the other hand, presented a different approach. It makes use of a Switch Divider Controller (Prospero) to allocate a subset of the physical switch resources into a Switchlet and makes this available to the control software through a propietary interface called Ariel. While working towards a similar goal, our MIBlet development has a different perspective. Interfaces defined in both BIB and Switchlet provide access to the underlying software entities, perform binding and manipulate "local" states. The MIBlet in this paper, however, provides a standardized view of the abstracted resources such that the management and control applications can manipulate the "local" states through a common interface. While both BIB and Switchlet have put enormous effort on defining their own interfaces, our MIBlet does not require any specific interface. It is applicable to any openly defined interface such as SNMP, CMIP, GSMP [11], etc. Both the BIB and MIBlet allow dynamic creation of state variable through creating new instances of interface and new entries in the information base, respectively.

Concerning resource sharing and policing, [3] proposes a simple "divider" which polices invocation on the Switchlet's Ariel interface to ensure that the controller does not utilize resources not allocated to it or in any way interfere with the operation of other Switchlets. To achieve this partitioning, the BIB uses the concept of binding. Binding refers to the activity of creating a requested service and entails the association of a set of network resources with the service management facilities. Our MIBlet shares a similar policing concept as Switchlet but enhances it with a more flexible and comprehensive sharing policy particularly in switching resources. Thus,

[4] This programmable node approach (with mobile code hosting environment) is also known as the discrete approach of "active" networks.

in this paper, we propose an intermediate partitioning scheme, which strikes the balance between static/hard partitioning and complete sharing.

With regards to organization of abstracted resources, neither BIB nor Switchlet clearly specify how it can be done even though their definition of interfaces may have some implications. Nevertheless, MIBlet uses a commonly accepted tree structure, which has been widely adopted in various standardization bodies for representing Management Information Base (MIB). The abstractions provided by MIBlet are therefore compatible and interoperable with existing and common management and control software.

6. Conclusions

In this paper we have discussed the virtual network (VN) concept as a means for organizing a programmable network. The management of VNs is organized into a network management layer and a resource management layer. At the network management layer, network-wide management and control functions are applied. For each VN, specific control and management functions can be applied, allowing for customized control and management of virtual network resources (VNRs). The resource management layer deals with the management of individual network resources. VNRs in a provider domain are represented by *Resource Agents*, which handle partitioning, and allocation of resources to customers. Resource Agents also host programmable controllers belonging to the various CNRM systems, enforcing customers' policies for the control and management of the resources allocated to these customers.

This paper particularly emphasized the design architecture of the Resource Agent. It focused on the partitioning function of virtual network resources through the provision of MIBlets. MIBlets are introduced as effective means to design VNs and to allow for customized control and management of these VNs. They result from a logical partitioning of MIBs to provide CNRM systems with abstract and selective views of the network resources. Static and dynamic partitioning schemes are combined to support VNs with different QoS requirements, and to allow for flexible resource allocation (e.g., by enabling network-level multiplexing). This paper has only studied the partitioning of ATM SNMP MIBs to experiment the VN structuring and the programmability of VNRM architecture using the ATM resources available in the NAL Laboratory. However, the MIBlet concept is generally applicable for partitioning any type of network resources. Indeed, all network devices are already endowed with MIBs containing large numbers of managed objects, which can be selectively abstracted to design VNs and to provide interoperable interfaces accessible by control and management software.

Based on the MIBlet concept, we have shown how we can effectively organize a network into VNs through simple, open and largely deployed MIB interfaces. MIBlets can be rapidly implemented without requiring new efforts for the definition and standardization of new dynamic binding interfaces. They can be deployed at a large scale in current networks, and future programmable networks more likely to be composed of programmable and non-programmable nodes. The MIBlet approach aims at capitalizing the effort already invested in the provision of standard MIB structures, protocols and interfaces, which allows interoperability with a variety of nodes, protocol stacks and management systems.

7. References

[1] Andrew Do-Sung Jun and Alberto Leon-Garcia, "Virtual Network Resources Management: A Divide-and-Conquer Approach for the Control of Future Networks", IEEE Global Telecommunications Conference (GlobeCom'98), Sydney, Australia, November 1998.

[2] Boutaba, R., "A Methodology for Structuring Management of Networked Systems", in IFIP Transactions, pp. 225-242, Ed. North-Holland, 1994.

[3] J.E. van der Merwe and I.M. Leslie, “Switchlets and Dynamic Virtual ATM Networks,” IM’ 97, pp. 355-368, May 1997.

[4] Z. Dziong, K-Q. Liao, and L. Mason, “Effective bandwidth allocation and buffer dimensioning in ATM based networks with priorities,” Computer Networks ISDN-Systems, vol. 25, pp. 1065-1078, May 1993.

[5] F. Kelly, “Effective bandwidths at multi-class queues,” Queuing Sys., vol. 9, pp. 5-15, 1991.

[6] G. Gallassi, G. Rigolio, and L. Fratta, “ATM: Bandwidth assignment and bandwidth enforcement policies,” Proc. GLOBECOM’ 89, vol.3, pp.1788-93, 1989.

[7] R. Guerin, H. Ahmadi, and M. Naghshineh, “Equivalent capacity and its application to bandwidth allocation in high-speed networks,” IEEE J. Select. Areas Comm., vol. 9, 1991.

[8] P. Joos and W. Verbiest, “A statistical bandwidth allocation and usage monitoring algorithm for ATM networks,” Proc. ICC’89, pp. 415-422, 1989.

[9] HungKei Keith Chow and Alberto Leon-Garcia, "Implementation and Performance Evaluation of ISAC: Integrated Service Internet with RSVP over ATM shortCuts," ICC'98, June 1998.

[10] Lazar, A. A., Bhonsle, S. and Lim, K. S., "A Binding Architecture for Multimedia Networks", Journal of Parallel and Distributed Computing, Vol. 30, No. 2, Nov. 1995, pp.204-216.

[11] "GSMP: General Switch Management Protocol", Ipsilon Networks, IETF RFC 1987.

[12] A. Ghalamallah and R. Boutaba, “Implementing a Distributed Web-based Management System in Java”, IEEE ITS/SBT’98, San Paulo, Brazil, 7-10 August 1998.

Biographical Note

Walfrey Wah-Fai Ng is an M.A.Sc. student at the University of Toronto. Andrew Do-Sung Jun and Keith HungKei Chow are Ph.D. students at the University of Toronto. Raouf Boutaba and Alberto Leon-Garcia are professors in the Department of Electrical and Computer Engineering at the University of Toronto.