MANAGEMENT OF ACTIVE NETWORKS

SYNOPSIS

This is a working document whose goals are the following.

a. Identify key issues and technologies required to manage active networks.

b. Develop strawman architecture for active network management.

CURRENT NETWORK AND SYSTEM MANAGEMENT

The overall architecture of a network management system, based on the *Simple Network Management Protocol (SNMP)* paradigm [4], is depicted in Figure 1.

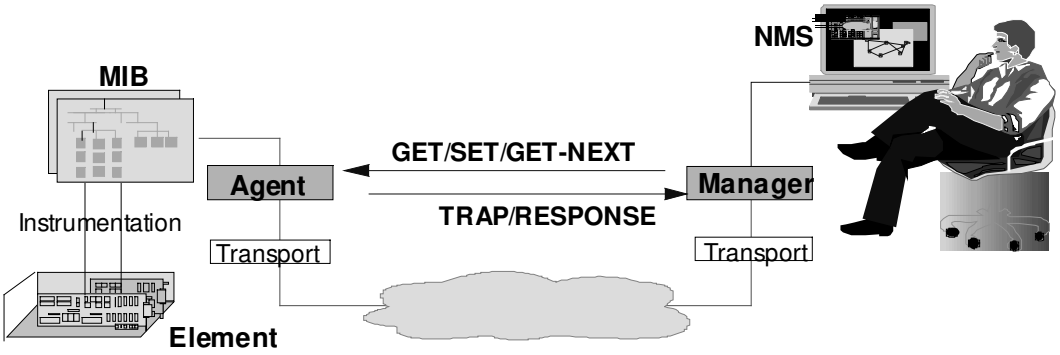

Figure 1: The Architecture of a Network Management System (NMS)

Network elements are *instrumented* to support (a) monitoring of element performance, (b) configuring element operational parameters, and (c) reporting exceptional operational events. Configuration and performance monitoring instrumentation is organized in a standardized naming directory called *Management Information Base (MIB)* [4]. The MIB provides a universal directory of names for configuration and performance data of network elements. An *agent* embedded in an element enables a *remote Network Management System* (NMS) to access and manipulate MIB variables at the element via the SNMP protocol. The SNMP protocol provides mechanisms to (a) read (GET) performance and status variables from an element MIB, (b) change (SET) configuration parameters, and (c) report events (TRAP).

In recent years, new management paradigm proposals tried to overcome some of the key deficiencies of the SNMP model. The *Management by Delegation (MbD)* [2] paradigm proposes a distributed hierarchy of managers. At the lower layers of the hierarchy, managers closer to the managed entities will be responsible for monitoring and controlling their operations. Managers in higher levels of the hierarchy oversee several managers to distribute the management duties. MbD is a very scalable proposition when compared to the model in Figure 1. The MbD paradigm can be extended to use mobile agent technology such as Java/RMI [1] or CORBA [3] to implement the protocol between the distributed managers, which are implemented as agents. The main impact of these paradigms is that the Manager box in Figure 1 becomes a distributed set of

manager agents communicating using RMI in the case of Java or IDL in the case of CORBA. The interaction between these managers and the managed object agent may still use SNMP.

For the discussion that follows, these differences in management paradigms will not overcome the management challenges introduced by active networks. Without loss of generality in the discussion that follows, we will refer to the model in Figure 1.

System management, while sharing several similarities with network management, involves a different paradigm. A system administrator has to handle configuration or problem management manually. In many cases, a system administrator uses generic remote access capabilities, such as telnet in Unix or custom applications/protocols in Windows NT, to manage remote systems, rather than a specialized protocol. Some management applications may use SNMP to access and manipulate element instrumentation. Often, however, configuration management functions are too complex to handle changes via SNMP and are thus accomplished by scripts executing at the element under the control of the NMS.

A large number of configuration management tasks are therefore automated via ad-hoc scripted tools. These scripts typically access and manipulate configuration data defined by respective files, rather than MIBs. Often, these scripts and management functions on which they depend are the result of a long-term ad-hoc evolutionary process that focused on management functions rather than systematic collection of instrumented data. This unstructured and ad-hoc manual approach to system management is a significant barrier to the efficient, secure and robust operation of active networks.

In summary, there are several commonalities with current practices in both system and network management.

1. **Mostly Manual Management**. Administrative staff manually handles a significant set of management functions. Some of these functions may be automated using AI based techniques. But there is still a lot of research required before fully automated network management is widely deployed.

2. **Centralized Management Applications**. Management application tools execute at centralized workstations to support manual management. The research community already proposed alternative models like Management by Delegation (MbD) [2] and Manager Of Managers (MOM) [5], but these are not yet widely deployed in real systems.

3. **Management by remote access to instrumentation**. SNMP supports remote access by applications to instrumentation data.

4. **Stand-alone Management**. Management functions and software are entirely separated from operating network software and applications.

5. **Static persistent management**. Managed components are assumed to be persistent relative to the time scale of their operational changes. For example, a MIB counter representing the number of packets handled by a component typically persists for a much longer time than the time between packets.

6. **Reactive Management**. Management of network is after network event ever occurred. Current management applications do not monitor and predict problems to take proactive steps.

For the discussion that follows, we use the generic term *active element* to refer to any active component in a node, such as an Execution Environment (EE), Active Application (AA) or component of an EE or AA. Notice that a delegation protocol or a capsule may be used to deploy components of an AA. It is also important to notice that we specifically envision using AAs as part of the ANM architecture as both managed entities and as components of the management software.

LIMITATIONS OF CURRENT MANAGEMENT TECHNOLOGIES IN HANDLING ACTIVE NETS

Consider the possibility of using SNMP-based management, as described above, to manage an active network. This requires the following.

1. Whenever an active element is loaded into an element, it is necessary to load respective instrumentation and MIB components and integrate these with the element management SoftWare (SW). It is also necessary to load similar MIB components into the NMS to enable management application tools to access this instrumentation.

2. Management application tools at NMS will have to be programmed to be able to process MIB data associated with active elements.

3. These dynamic changes in elements management SW and the NMS will have to be synchronized and coordinated with the dynamic changes of the active network.

4. MIB structures at element and NMS will have to change dynamically on a time scale similar to the time over which MIB variable change. Furthermore, MIB data may have to persist even if the respective active element has terminated, to facilitate analysis of potential problems or recovery of configuration states.

5. With each active element SW, it will be necessary to create respective instrumentation SW and MIBs to handle the respective management functions.

6. An active component functions in a dual role of both, a network element and as a local system element. As such it needs to be managed both, in its role as a network element, as well as in its role of accessing and operating on local system resources. This requires unification of network and system management functions.

7. It should be possible to allow newly developed system-level software to reuse existing system/network management paradigms. Thus, a major challenge that must be met for managing dynamic and ever-evolving networks is to extend the adaptability of the network services to their management. Statically defined network management (NM) databases of the style of SNMP MIBs can no longer be used to specify the management of evolving systems because this would require constant updates to the MIBs and NM tools to reflect the additions of new entities.

Additionally, active elements typically need to adapt to and even control the network behaviors. They must thus be able to access data concerning network performance and configuration, as well as effect configuration changes to control network resources. Therefore, in contrast with traditional network applications, which are entirely separated from management software, active applications will need to integrate monitoring and control capabilities. The traditional approach to network SW design has been to incorporate function-specific monitoring and control capabilities with every protocol/network-system. Thus, for example, routing SW uses a routing information protocol to monitor network topology information, and transport-layer SW can use RSVP to control allocation of resources. Each such application requires its own specialized instrumentation and access protocol to facilitate monitoring and control functions.

The approach of incorporating monitoring and control functions -- including specialized instrumentation and access protocol -- with each application, does not scale efficiently for a large number of active applications SW with a broad variety of adaptation functions. Instead, a more reasonable approach is for active applications to share instrumentation and access mechanisms to monitoring and control functions. Therefore, active networks can greatly benefit from a shared monitoring and control SW infrastructure, including instrumentation and access mechanisms. Thus, in contrast with current networks, active networks require an integration of management mechanisms and application SW.

Clearly, the management framework offered by SNMP is unable to handle the management needs of active networks. In what follows, we identify issues and requirements for active network management mechanisms.

ISSUES & REQUIREMENTS FOR MANAGING ACTIVE NETWORKS

This section aims to describe substantially novel technologies required to manage active networks. It derives, in part, from the considerations discussed in the previous sections. Notice the use of the word "should" rather than "shall" in the requirements below; this intends to reflect goals rather than strict requirements; the prefix G thus stands for "Goal".

G1: Dynamically Composable Management
ANet management should provide means for dynamic composition of management modules, to adapt to dynamic changes in active elements of the network. In particular it should:
(a) Support dynamic adaptation of node instrumentation and management SW to reflect changes in active elements SW and composition with other node management SW.
(b) Admit similar dynamic adaptation of management SW at NMS to reflect changing configuration of active elements SW.
(c) Coordinate changes and persistence of management SW at nodes and NMS with changes in active element configurations.
(d) Provide means for sharing inter-EE functionality, that is, provide mechanisms for EEs to export their interfaces to the NM so that other EEs can discover and use these interfaces for their operations.

G2: Backward Compatibility With SNMP
ANet management needs to support the SNMP management framework where possible. In particular, it should:
(a) Be incorporated with active elements instrumentation for configuration and monitoring.
(b) Provide appropriate MIB extensions to access this instrumentation via SNMP.
(c) Provide means to dynamically incorporate these management SW components at elements and at network management stations (NMS) in coordination with the execution of the respective active elements.

G3: Applications-Controlled Management
ANet management should provide means for active applications to monitor and control network configuration and behaviors. In particular, it should provide mechanisms for active applications to:
(a) Configure network topology and resources as needed to support their needs.
(b) Obtain network topology and resource availability information.
(c) Monitor network performance parameters.
(d) Monitor exceptional network performance events.

G4: Automation of Configuration Management
ANet management should provide means to facilitate automated configuration changes and to assure the operational consistency of resulting configurations and recoverability of previous configuration states. In particular ANet management should:
(a) Provide means to undo configuration changes, whether resulting from actions by AAs, through management SW, or even from attacks, to recover an operationally valid configuration state.
(b) Provide means to assure configuration consistency through changes.
(c) Provide means to detect violation of acceptable configuration states to protect against attacks.

G5: Automation of Problem Management
ANet management should provide means to automate detection, isolation and handling of network problems.

G6: Providing Semantically Richer Network Data

ANet management should maintain and provide a semantic-rich data model of the ANet, to facilitate automation of configuration and problem management functions. This model should include, in addition to raw configuration and performance data, relationships that can influence problem and configuration behaviors, event correlation knowledge and configuration consistency knowledge.

G7: Generation of Active Element Management Data & Instrumentation
ANet management should provide means to derive management instrumentation and data models from the structure of Execution Environment (EE) and/or Active Application (AA) code.

G8: Secure Management
ANet management should provide secure access to management capabilities.

G9: Proactive Management
ANets must be able to predict some network failures/problems (e.g., congestion). Managed objects should be self-monitoring and take steps to detect and to rectify abnormal behavior. E.g., by extending the MIB to gather more information and/or by applying automated reasoning technologies.

ACTIVE NETWORK MANAGEMENT (ANM)

As explained above SNMP offers several limitations for active networks. We define a new type of NM paradigm especially designed for highly dynamic and composable networks: the Active Network Management (ANM) framework.

ANM ARCHITECTURE

Figure 2 depicts the overall architecture for ANet management. The ANet Node Manager consists of SW to monitor, configure, analyze, and control a node. The Node Manager interacts with local node instrumentation (via the NodeOS API) to access performance data, configuration functions and operational events. It interacts with EE's to support (a) management of the EE configuration, problems, and performance; (b) adaptation of management SW to dynamic changes in active applications; and (c) management of node configuration objects by other EE's or AA's. The Node Manager exposes APIs to the EE to enable active applications to adapt and control network resources; active applications can monitor network performance and configure network resources. The Node Manager interacts with the NMS to support remote management functions. In particular, the Node Manager interacts with the NMS to adapt its SW to dynamic changes in the active applications.

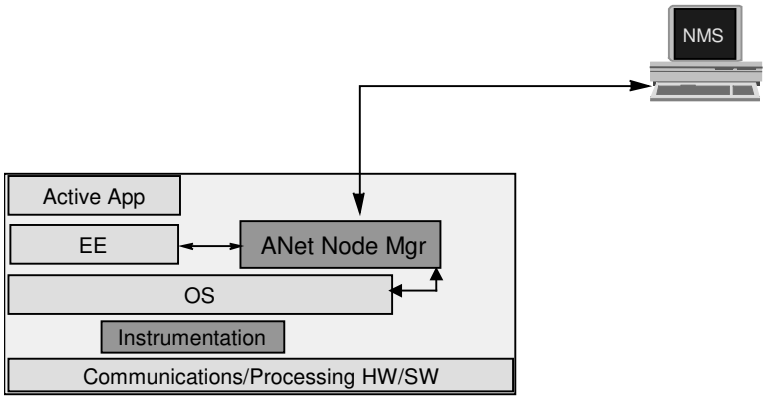

Figure 2: Overall architecture of ANet Management

Figure 3 depicts a more detailed view of the Node Manager Architecture. There are two forms of interactions among management modules (a) synchronous access, depicted by black arrows, and (b) asynchronous interactions to process network events, depicted by red arrows. The Node Manager is organized essentially as a three-layered architecture. The *instrumentation layer* at the bottom provides instrumentation adapters to support access to event and management data provided by various node components. The *Active MIB (AMIB)* in provides access to this instrumentation. The management *Data-Modeling Layer (DML)*, in the middle, organizes management data to enable manager applications to access and analyze data models of the network configuration and performance behavior. The DML handles both synchronous data access by applications, as well as asynchronous event notifications.

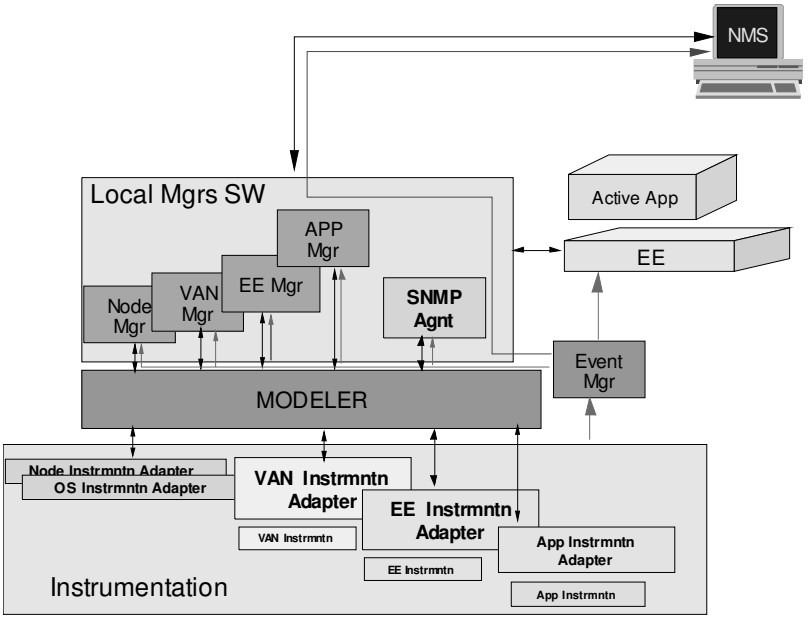

Figure 3: Architecture of the Node Manager

The Local Manager Software (LMS) is populated with local managers that are responsible for managing specific node functions and components, including communication and OS elements, and components of Virtual Active Network (VAN), EEs, and AAs. Additionally, the local managers support access mechanisms and protocols to interact with the NMS. In particular, it supports SNMP agent to access data and events of the instrumentation layer. It can also support an HTTP server to support Web access to local managers and to the data-modeling layer.

AUTOMATING MANAGEMENT OF ACTIVE COMPONENTS

The primary goal of this section is to define a mechanism that will enable automation of the management of active components (EEs or AAs) without implicitly prescribing a particular NM paradigm or composition method. We propose a Common Management Framework (CMF) that allows such diversity and power without compromising performance or simplicity.

COMMON MANAGEMENT FRAMEWORK (CMF)

The CMF design is targeted at coarse grain management of EEs. The finer grain management of the components within EEs and their composition may be indirectly supported by the CMF, as we will see later, through specific instantiations of the CMF design.

To support multiple NM frameworks, we introduce a discovery mechanism to probe newly deployed EEs. The idea is quite simple and is somewhat similar to the approach followed today on the Web. Each network service listens on a known port. After deployment, the EEs respond to a predefined and universally agreed-upon ANEP packet **INIT** (the equivalent of GET / in HTTP). The **INIT** command will cause the EEs to respond with am ANEP packet **EXPORT** that includes in the payload a MIME-encapsulated reply. In general, the **EXPORT** packet contains information to be used for the configuration and monitoring of the EE itself and the configuration of other related EEs.

Using MIME encapsulation as three main advantages:

- It allows the reuse of existing powerful tools, such as HTML browsers, that can be extended to handle user-defined application types and that conveniently integrate into current Web-based technology.

- By breaking down a potentially very large application domain for managing active networks, it makes the problem more tractable. In particular, it allows the design of domain-specific semantic interpretation of the exported interfaces without the need of extremely general and potentially ambiguous specifications.

- The MIME dereferencing mechanism naturally supports extensibility.

The specification of each MIME could be facilitated by using XML to offer the EE developer available XML support tools in generating the grammars. Later we describe some MIME types that can be a starting point for the CMF and assume that their syntax be automatically derived by and XML DTD specification.

In general, the **INIT/EXPORT** mechanism can be seen as the bootstrap mechanism that allows an EE to integrate itself into the NM infrastructure. For example, in a simple scenario, the service replies with an encapsulated message in HTML format. This reply format allows the administrator to use standard HTML forms to configure and later interactively monitor the EE through HTML at the NMS. Other more complex NM paradigms can also be naturally supported. For example, in paradigms that focus on providing composibility, the export operation could cause the installation of EE-supplied methods to be added into a common brokerage system in a manner similar to the CORBA model.

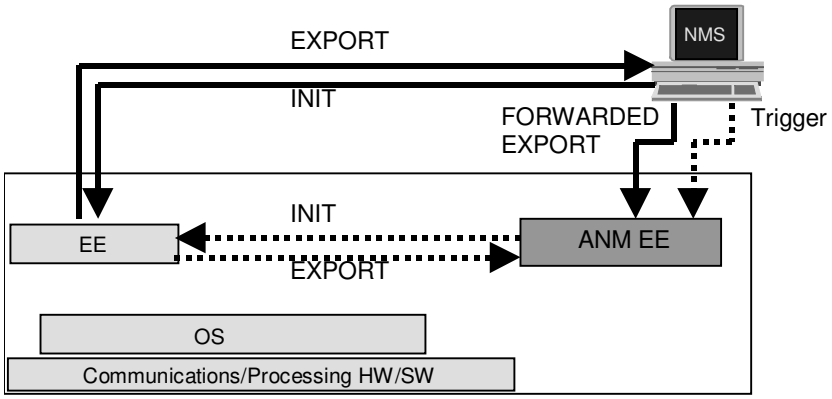

Figure 4: CMF flexible runtime adaptation

In general, the **EXPORT** packet will be returned to the remote NMS that deployed and later queried the service. In some cases (like an HTML reply) these export operations will be directly processed by the NMS for remote management. In other cases, this export information may be forwarded to other appropriate entities located in the network. A common case will be the one in which the exported interface will be forwarded to a NM EE residing on or near the same node where the EE was deployed (see Figure 4). Thus, after having received the reply from the ANEP INIT command, the NMS would forward the reply back on

the node to the NM EE. The NM EE will then process the export operation from the deployed EE and augment and extend the local NM system to incorporate the new arrival in its NM scope. Yet another possible scenario would allow the NMS to send a **Trigger** event to a NM EE to send an **INIT** packet to a newly deployed EE. In this case the EE would respond directly to the NM EE thus avoiding the forwarding step.

The next sections shows (1) how CMF can be used to activate legacy SNMP-based solutions and (2) an ad-hoc NM paradigm specifically designed to support fine-grain, dynamically composable active network services.

AN SNMP MIME AS A BACKWARD COMPATIBLE INTERFACE

This example shows how our framework can be used to build a bridge with traditional non-active NM approaches. It is important to keep backward compatibility so that one can leverage legacy applications in solving problems that do not require innovation. For this reason, we show how an SNMP MIME type and its corresponding applications are capable of dynamically deploying and configuring SNMP agents. A deployed service wanting to use SNMP replies with information encapsulated in an SNMP-specific MIME type (application/SNMPV2). The reply might be similar to the following:

content-type application/SNMPV2

<SNMP>
<AGENT=http://www.abone.org/SNMP/snmpd/>
<PORT=8000/>
<MIB=http://www.abone.org/SNMP/mib.txt/>
<ACL=http://www.abone.org/SNMP/acl.conf/>
<PARTY=http://www.abone.org/SNMP/snmpd/party.conf/>
<VIEW=http://www.abone.org/SNMP/snmpd/view.conf/>
<CONTEXT=http://www.abone.org/SNMP/snmpd/context.conf/>
<ENVIRONMENT VAR=MIBFILE VAL=mib.txt/>
</SNMP>

This reply, which would be processed by the NMS, specifies the URL of the SNMP agent, the associated MIB, and access control information to be loaded with the service. Following this reply would cause the NMS to deploy and configure the SNMP system (the agent snmpd, the client and the server's MIB and the configuration files). The deployment of the agent and the associated configuration files could be easily carried out through Anetd. In addition, the EXPORT reply would also cause the management station to either spawn an SNMP browser to access the deployed service or update an existing SNMP management application with the new MIB.

ADAPTATION TO ACTIVE CODE

Active elements are deployed dynamically, changing the contents of a node. Network management components will need to adapt to these changes as follows.

- The instrumentation layer or AMIB needs to incorporate components to configure and monitor active elements; these components need to be updated to reflect the deployment of new active elements or deletion of those. One proposal is for an active element to carry its MIB instrumentation and deploy it in nodes as related elements. Another proposal (which can be combined with the previous one) is to create a uniform AMIB that is deployed in an EE/AA-independent manner to instrument configuration, problem, and performance management of all active elements.

- The DML will need to incorporate respective managed object models representing active elements; the DML needs to be updated dynamically and bound dynamically to respective components at the instrumentation layer.

- The LMS, associated with a set of active elements will need to be updated dynamically to support configuration and problem management of the respective active application.

We outline here a mechanism and an initial specification of how this runtime adaptation may take place. After deployment, an EE that subscribes to the CMF may be queried with the ANEP **INIT** command and return a MIME-encapsulated reply of the form:

content-type application/ANM

<AMIB>
new objects to be added to the active MIB
</AMIB>
<DML>
new abstraction functions to be added to the DML
</DML>
<LM>
a new local manager to be loaded into the NM EE as an AA on b
</LM>

The following sections describe in more detail key design challenges in the three layers of the ANM architecture and give a more detailed syntax for their runtime adaptation.

The Instrumentation Layer (AMIB)

As new EEs or AAs are added to a node they may export an instrumentation to be used to access their services and/or performance metrics. Such information will be exported to an extensible Active MIB (AMIB) that will be automatically extended upon the export operation. The AMIB will have a structure very similar to the one of current SNMP MIBs with two key additional features.

1. Each variable will have a specific method and calling convention associated to it. The instrumentation layer will use this method to access the information provided by the associated network service.

2. Each variable will have several general-purpose attributes defining specific characteristics of the variable to be used for regulating polling modalities, access restrictions, synchronization requirements, etc.

The instrumentation layer will be responsible for correctly interpreting the exported MIB variables and for setting up the references to the methods so that the variables may be transparently accessed by the data-modeling layer or directly by the LMS.

The sample syntax to export AMIB variables is the following.

<AMIB>
    <EXTEND MIB=*url*>
    <OBJECT *name*>
        <RPC=*rpc*/>
        <MODE=*access*/>
        <ATTRIBUTE1=*value*/>...
    </OBJECT>
</AMIB>

This export operation allows (1) to add a new object to a base AMIB specified in *url*, (2) to specify its fully qualified name *name*, (3) prescribe an access mode *access* and (4) define a number of application specific attributes for the objects.

For example:

```
<AMIB>
        <EXTEND MIB=http://www.abone.org/amib/mib.txt>
        <OBJECT service.protocol.rsvp.Encapsulation>
                <RPC=get_version()/>
                <MODE=READ/>
                <CRITICALITY=0 FREQUENCY=0 ACCESS=0/>
        </OBJECT>
        <OBJECT  service.protocol.rsvp.Encapsulation>
                <RPC=get_encapsulation() />
                <MODE=READ />
                <CRITICALITY=0 FREQUENCY=0 ACCESS=0/>
        </OBJECT>
        <OBJECT NAME=service.protocol.rsvp.RefreshInterval>
                <RPC=get_refresh()/>
                <MODE=READ />
                <CRITICALITY=0 FREQUENCY=0 ACCESS=0/>
        </OBJECT>
        <OBJECECT service.protocol.rsvp.SessionTable>
                <RPC=get_table()/>
                <MODE=READ/>
                <CRITICALITY=0 FREQUENCY=1 ACCESS=0/>
        </OBJECT>
        <OBJECT service.protocol.rsvp.BadPackets>
                <RPC=get_badpackets/>
                <MODE=READ/>
                <CRITICALITY=1 FREQUENCY=1 ACCESS=0/>
        </OBJECT>
</AMIB>
```

The above reply might be returned by an activated RSVP service such as ARP. This reply exports management operations and indicates that these operations should be used to extend a base AMIB located on an ABONE code server. The object name specifications will be automatically added to the ABONE active MIB, thus actively extending it at runtime.

The example also includes the following attributes that convey useful additional information.

- *Criticality***.** This attribute specifies to what extent the value of the object is critical for the correct operation of the system. For example, (0) means that the value of the object does not affect operations, (1) means that the value of the object has significance at the local level, and (2) means that the value of the object has significance at the system level.

- *Frequency of Change.* This attribute tells the NM system how often the value(s) of the object is expected to change. For example, (0) means that the value(s) of the object does not change, (1) means that the value(s) of the object changes with the frequency of human interactions (every seconds to minutes), and (2) means that the value(s) of the object changes with the frequency of system interactions (less than every second).

- *Accessibility.* This attribute regulates who can access or affect the objects. For example, (0) means that the object is public and anyone can access it, (1) means that the object is accessible by authorized users and that access control should be enforced, and (2) means that the object should be accessible only by the super-user.

These attributes facilitate the runtime inclusion of new objects into an NM infrastructure. For example, if a new monitoring service is added to an active network node to perform a specialized monitoring function, this service may introduce new objects in the hierarchy upon deployment and classify them according to their significance. Depending on what the monitor is doing it would associate different values to its attributes. For example, if the monitor produced an object that estimated CPU utilization, its attributes

would probably be set to 0, 1, and 0 to indicate that this object has a non-critical significance, it changes every few seconds, and it should be widely accessible. On the other hand, consider a monitor for intrusion detection that produced intrusion attempt alarms. For obvious reasons, this object would probably be characterized as 2,1,1 meaning that the object interpretation is critical, it should be checked every few seconds, and only privileged users should access it.

The Data Modeling Layer (DML)

Upon deployment of an EE or additions of new objects in the instrumentation layer, new modeling services may be adopted. These services will provide access to higher level of abstraction than the low-level AMIB objects. The functions contained in this layer can therefore be regarded as available functionality to the EEs in the node or the NMS. The addition of these methods will be expressed by the following syntax.

```
<DML>
        <METHOD  name>
                <ARGUMENT= type/>
                <URL= url/>
                <COMMENT= description/>
        </METHOD>
</DML>
```

Where *name* is the name of the function, *type* specifies the argument(s) type, *url* is the location of the code that implements the function (to be loaded through an RMI-like mechanism), and *description* is used to facilitate the adoption of the functions by third parties.

An example of how this DML export operation could be used is given below:

```
<DML>
        <METHOD  Get_all>
                <URL=http://abone.org/getall.class/>
                <COMMENT="The function Get_all returns the entire MIB tree if mode is 0 the list is in depth-first, if
                mode=1 the list is in breadth-first"/>
        </METHOD>
        <METHOD get_topology>
                <ARGUMENT=integer/>
                <URL=http://abone.org/gettopology.class/>
                <COMMENT="The function Get_topology("n") returns the list of directly connected neighbours in overlay
n"/>
        </METHOD>
        <METHOD Get_routes URL=http://abone.org/getroutes.class/>
                <ARGUMENT=integer/>
                <COMMENT="The function Get_routes("n") returns the routes in the overlay "n"/>
        </METHOD>

</DML>
```

The method Get_all() implements a basic abstraction function that returns the entire MIB. More interesting and useful functions like Get_topology(n) or Get_routes(n) could be installed to offer abstraction to be directly used by EEs to access topology and routing information on the local node.

Local Manager Software (LMS)

The managers can also be deployed and configured at runtime. The managers can be seen as AAs running within the NM EE. The instantiation of these managers will follow a syntax similar to the one used for the DML. The general syntax used to specify the installation of a local manager is the following.

```
<LM>
        <LOCAL MANAGER manager_name>
                <COMMAND-LINE ARGUMENT=arguments/>
                <URL=url/>
                <COMMENT=description/>
```

</LOCAL MANAGER>
</LM>

ANET MANAGEMENT DESIGN ISSUES

THE DATA MODELING LAYER (DML)

The goal of the Data Modeling Layer (DML) is to organize management data in a form that enables local and remote manager applications to access and analyze the network configuration and performance behavior. The DML data models should enable managers to configure and control network components and to monitor and correlate their behaviors. The DML is distributed among network nodes and contains appropriate support for replication and caching to facilitate rapid access to network management information.

The DML shall provide the following management information models:

1. Network topology model; including topologies of the physical layer, IP layer, VANs and relationship between these layers.

2. Node elements operating models; including node HW and SW components instrumentation, and active elements operating at a node.

The DML data model shall include descriptions of managed objects as follows:

1. Configuration attributes of managed objects.

2. Performance behavior attributes of managed objects.

3. Relationships of managed objects to other managed objects.

4. Events associated with managed objects.

Additionally, the DML will include data and knowledge models to facilitate automation of problem management, configuration management and performance management.

The key DML design issues are the following.

1. How to develop and organize data models of network components that facilitate automation of management functions.

2. How to distribute DML among multiple nodes and support efficient access to DML data through replication and caching.

3. How to support rapid dynamic changes in DML to represent changes in active elements deployment.

LOCAL MANAGER SOFTWARE (LMS)

A local manager is responsible for managing node functions under local and/or remote control. Local managers are also installed dynamically and implement distributed NM services. These managers may

access either the DML functions or the AMIB directly and are the basic building blocks of the Active NMS. We have identified some necessary and very important local managers.

Node Manager: Manages local HW, SW and bandwidth resources; it configures and monitors the performance of these resources and handles their operational problems.

VAN Manager: Configures and monitors the performance of VANs. This includes creating and configuring a VAN, allocating local resources to VANs, monitoring performance and problem behaviors of a VAN and recovering from these.

EE Manager: Configures and monitors the performance of an EE. This includes configuring the EE, allocating resources to an EE, linking EE with VANs, and monitoring the performance of an EE, monitoring active elements executing on top of an EE.

Active App Manager: Can be developed to configure and monitor individual active applications.

MULTI-DOMAIN MANAGEMENT

ANet management should operate across multiple administrative domains. This will require the development of policies and mechanisms for authorization and security.

INTEGRATION WITH SNMP

ANet Management integrates with SNMP in two forms. First, the Data Modeling Layer (DML) uses SNMP instrumentation of a local node to configure and monitor it. Second, the DML and local managers can export their interfaces via an SNMP proxy agent to enable remote SNMP manager to access to local management capabilities. This will require development of new MIBs to support common SNMP instrumentation and naming of ANet local management capabilities. This architecture is depicted in Figure 3.

VAN MANAGER

VAN is a unit of organizing end-end connectivity among a group of node activities, and virtualizing the underlying network resources. The VAN management functions, supported by the VAN manager, include the following.

- Creation and termination of a VAN; supporting joining/departure of nodes from a VAN.
- Allocation of bandwidth, processor, and memory resources at nodes to support the VAN processing.
- Linking VAN nodes via respective virtual links.
- Binding active elements to VAN nodes to support their end-end connectivity needs.
- Providing security of VAN activities and interactions.
- Monitoring VAN performance, detecting and recovering from operational problems.

REFERENCES

[1] Ken Arnold and James Gosling. The Java™ Programming Language, Second Edition, Reading, MA, Addison-Wesley, 1998.

[2] Germán Goldszmidt and Yechiam Yemini. "Distributed Management by Delegation," in the 15th International Conference on Distributed Computing Systems, IEEE Computer Society, Vancouver, British Columbia, Canada, June 1995.

[3] Jon Siegel. CORBA Fundamentals and Programming, John Wiley & Sons, 1996.

[4] William Stallings. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Third Edition, Reading, MA, Addison-Wesley, 1999.

[5] Douglas W. Stevenson. "Network Management: what it is and what it isn't," White Paper, April 1995, http://netman.cit.buffalo.edu/Doc/DStevenson/.