Service Configuration and Management in Adaptable Networks *

Livio Ricciulli

Computer Science Laboratory, SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025, US
livio@csl.sri.com
http://www.csl.sri.com/ancors

Abstract. We describe ANCORS, an architecture for the design, configuration, and management of adaptable networks. We describe the primary components of the architecture and their common system management infrastructure. We describe alternative techniques that can be used for the management of adaptable networks and discuss their relative strengths. We then propose an open and extensible management framework for adapting the management infrastructure to newly deployed network services. We exemplify the use of this framework by outlining four different representative management applications.

1 Introduction

Current networking systems are very static and are a result of years of standardization efforts that allow different vendors and software developers to interact through a set of well-defined protocols. Active networking is motivated by the notion that the improvement and evolution of current networking software is greatly hindered by slow and expensive standardization processes. Several active networking research projects [1, 4, 5, 12–14, 11] try to accommodate changes to network software by facilitating the safe and efficient dynamic reconfiguration of the network. Adaptive networks may be seen as the composition of the two main orthogonal approaches to active network design discussed in [13].

- In the *discrete approach* administrators issue explicit commands that load, modify, or remove networking software. With this approach a network is active in the sense that it can be dynamically changed administratively.
- In the *integrated approach* the network is modified by the data packets that travel through it. When packets travel through the network, they automatically cause required software resources to be loaded on demand. This approach is being followed today by most active networking research and allows a much finer-grain dynamism.

* The work presented in this paper is currently funded by the Information Technology Office of the Defense Advanced Research Projects Agency, under contract number DABT63-97-C0040.

Adaptable networks result in much more flexibility in designing and using networks, but pose several challenging technical problems. One of these problems is how to provide a unifying paradigm for the management of active networks. As new active network software is developed and deployed, both its static characteristics and runtime behavior should be made known. It should be possible to allow newly developed system-level software to reuse existing system/network management paradigms and delegate its management and monitoring functions to them. 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 Management Information Bases (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.

We are exploring new ways by which network management can be dynamically updated to include new services in its control and monitoring scope. Our NM paradigm focuses on supporting services that are relatively permanent and long-lived and that can benefit from having a specialized monitoring and configuration infrastructure. Examples of these kinds of services are the execution environments (EEs) produced by current active networking research groups, engineering and prototyping tools such as the one described in [10], and network monitoring tools like RMON and intrusion detection engines from the EMERALD project [7].

2 Overall Picture (ANCORS)

The ideas in this paper are derived from the ANCORS (Adaptable Network COntrol and Reporting System) project. ANCORS is intended to streamline and, at the same time, enrich the management and monitoring of adaptable networks, while adding new support to the NM paradigm to assist network designers.

ANCORS targets an active network environment, where powerful design and assessment capabilities are required to coordinate the high degree of dynamism in the configuration and availability of services and protocols. To this end, we have formulated an architecture of a network management and engineering system that, while inheriting some components from current NM technology, introduces distributed simulation as an additional tool for design and performance assessment. Some components of the ANCORS architecture map very well to already existing technology. Recognizing this, the architecture has been explicitly designed to accommodate other NM engineering solutions.

The ANCORS architecture is divided into data, assessment, and control layers, as illustrated in Figure 1. The data layer operates at the data packet level and offers a set of services for the manipulation of network data. The assessment layer performs analytical reviews of network behavior to extract relevant semantic information from it. The control layer performs higher-order functions

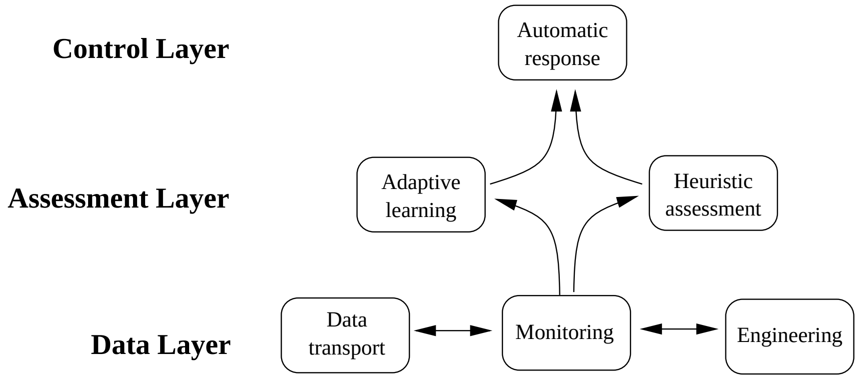

Fig. 1. The ANCORS Architecture

based on expert knowledge. All the components constituting these logical layers may be independently deployed throughout the network, using common system management support. ANCORS may distribute data-layer services across domains[1], but deploys assessment-and control-layer services to the specific domain they manage. Depending on the amount of resource sharing resulting from the deployment of active networking services, the assessment layer may also be distributed across multiple domains. Because the control layer must possess a significant amount of authority to perform changes in the network, it should be deployed only within a single domain. Several control services may then cooperate at the inter-domain level to exchange information for making better control decisions about their respective domains.[2]

3 Deployment and Management Infrastructure

Active network services require similar support functions from network or system management. The support functions can be broadly characterized as those achieving (1) process control, (2) configuration, or (3) monitoring.

Process control functions allow the loading and unloading of network services to and from network nodes. The physical location of the code that implements the network services may be different from the physical location of their deployment. For this reason, a reliable transport protocol such as TCP should be used to transfer the code. We have implemented a software prototype (*Anetd* [9]) that performs these process control functions. The prototype has a very flexible and general-purpose API to deploy and control (1) native executables, (2) Java applications, and (3) ANCORS executables.

[1] In this context, a domain consists of a collection of software and hardware objects managed by a single administrative authority.

[2] A discussion about inter-domain information exchange between control service is beyond the scope of this paper.

Configuration functions write control data into the network services after they have been loaded onto the intended nodes, to integrate them into the network node and possibly tailor them to particular needs.

Monitoring functions result in reading data from the network service to supervise its operation. This support function may be invoked remotely as in SNMP and CMIP, or locally by monitoring agents like RMON, to collect performance data.

3.1 Process Control

Anetd is a system management facility for managing the deployment, operation, and monitoring of deployable network services in an active network. *Anetd* is specifically targeted for the secure management and coordination of active networking research over the Internet. *Anetd* follows the discrete active networking approach, providing code mobility to legacy network software (e.g., SNMP agents) and system management support for new active networking applications. *Anetd* support focuses on services that are fairly permanent and long-lived, that can benefit from having a separate system management infrastructure, and that are fairly encapsulated (i.e., they do not rely on large numbers of shared libraries that may not be commonly available). *Anetd* also facilitates the deployment of auxiliary resources needed by the applications to ease the porting of existing software. However, *anetd* is not intended to deploy a large number of libraries or require large installation directories.[3]

From a system management perspective, *anetd* views all services within the ANCORS architecture (Figure 1) as equivalent. *Anetd* handles process control requests coming from the management stations or automatic response services to either load new services or terminate existing ones. Its placement in the ANCORS layered architecture [11] is illustrated in Figure 2.

The assessment layer interprets monitoring results from the data layer, and the control layer reacts to significant conditions as they are reported by the assessment layer. The automatic response services may reconfigure both the assessment services and the data-layer services in response to changes in the network behavior. Note from Figure 2 that this architecture allows the traditional but nonscalable approach of having the management station directly monitor and control the data layer. This aspect of our architecture can be very useful when simple network management technologies are employed that do not require ad-hoc distributed monitoring functions (e.g., when using legacy SNMP-based polling mechanisms).

3.2 Configuration and Monitoring (Network Management)

Because in most cases the configuration and monitoring functions are intimately tied to the semantics of a network service, in active networks, these functions

[3] In these cases, we advise standard manual installation techniques or (as in the case of Java) bundle the required resources with *anetd* in advance.

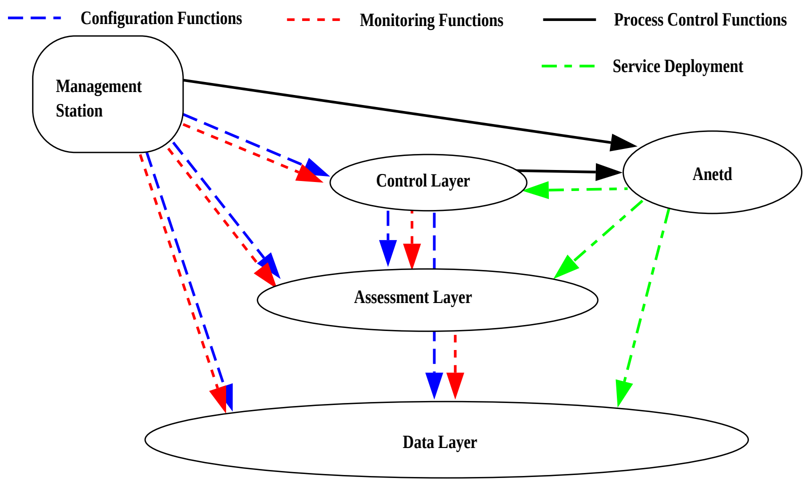

Fig. 2. ANCORS's Management Architecture

cannot be generally known in advance by the NM tools. As new services are deployed, the NM tools should adapt to the newly introduced functionalities. Two possible approaches can be followed in providing the management of active networks. We believe that both approaches should be allowed and should be integrated in a common management framework that can make best use of their respective advantages.

Active Management One strength of active networking concepts is that the network can be adapted at runtime by network operators. Within this framework, when a new network service is deployed, all necessary management agents and control software can also be deployed by the network operators in parallel. For example, as a new admission control protocol is deployed on the routers of an IP network, its corresponding MIB and SNMP agent may also be dynamically deployed. Such adaptation can also be achieved at a finer granularity by just adding methods to an existing agent that can access the new deployed service (as suggested in [2]).

Knowledge-Based Management Knowledge-based management (KBM) tries to integrate new network services much more closely into the NM system to provide greater software reuse and composition. In this paradigm when a new network service is deployed, it only exports an interface without requiring the deployment of specific NM code. In this context the NM management software should be able to correctly interpret the exported interface and directly integrate the new service in a pre-existing semantic framework. For example, when a new service is deployed it may export several alarms indicating failures and/or methods through which the service could be monitored or configured. The methods and alarms would then be interpreted and categorized by an existing NM application into specific predefined classes. For example, a service providing access control and bandwidth allocation such as RSVP may export an alarm indicating that the provisioned bit rate is not available to the service or there is an unusual number of bad packets. In KBM such alarms would be correctly interpreted and perhaps correlated with alarms originating from other services. In summary, KBM supports a higher level of composition through the dynamic extensibility of NM application.

Discussion KBM is hard because it requires the design of predetermined semantic contexts that are general enough to be extensible but that are not too general to be impractical to implement. However, it provides a logical framework through which one could achieve a high level of integration and more software reuse.

Active management is conceptually simpler and perhaps more practical but lacks the ability to support composition. When a new service and its corresponding management code are deployed, they could be (and in some cases should be) completely isolated from the rest of the NM system. For instance, in the above example, when an alarm is generated as a result of unavailable bit rate, it must be interpreted by a human and manually relayed to the service provider in the form of a complaint.

It is not clear which approach is preferable. In general, active management may be preferable for active applications that do not have much interaction with the system or other active components, while KBM will provide support for composition of services having a high degree of interdependencies. We believe that the management of programmable networks should allow both approaches and should be designed in an open fashion.

In the next sections we describe a design of a powerful and extensible common management framework (CMF) that tries to merge these two concepts through the reuse of Web-based tools. We will further exemplify KMB in Section 3.2.

4 CMF

An infrastructure to manage and support active networks must be open, simple, and flexible, and therefore we do not intend to have a unique format for the information exchanges between network services and management stations.

To support multiple management frameworks, we introduce a discovery mechanism to probe newly deployed services. The idea is quite simple and is somewhat similar to the approach followed today on the Web. Each network service listens on a port known to *Anetd*. After deployment, the services respond to a predefined and universally agreed-upon *Anetd* command **INIT** (the equivalent of GET / in HTTP). The **INIT** command will cause the network services subscribing to CMF to respond with a MIME-encapsulated reply. In general, the reply contains information to be used for the configuration and monitoring of the service itself and the configuration of other related services. In other words, as the result of an **INIT** command, the services export the interfaces necessary for their operation.

Using MIME encapsulation as three main advantages:

1. 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.
2. 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.
3. The MIME dereferencing mechanism naturally supports extensibility.

In the simplest 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 service through HTML forms.

5 CMF Applications

We will describe the design of four different management APIs that are being developed as part of an ongoing effort in realizing a worldwide testbed for active networks (ABONE [8]). These examples show how we intend to use our CMF design in enabling active networking software to be included at runtime into an active network management infrastructure. We refer to network management, in this case, as the ability to monitor and control aspects of the active networking software that go beyond the process control functions of *Anetd* and allow much finer grain resolution within a deployed service.

5.1 The text/ancors MIME

Within the *Anetd* implementation of CMF, we have developed a specific MIME environment (*text/ancors*) that assists in the deployment, monitoring, and control of active networking software. This MIME, can be used to (1) generate GUIs through HTML forms or (2) to build, in command-line mode, ad-hoc management front ends based on standard Unix string processing tools like *Perl* or *Awk*, or to operate from terminals that do not support graphics.

Each of the lines returned by a deployed network service can be either a simple text line or a GUI line. Text lines and GUI lines are differentiated by the fact that a GUI line starts with tags LOAD, QUERY, INIT, CONF, GET, PUT, KILL, GETACL, or GETWEB (that correspond to *Anetd*'s commands), while text lines do not. GUI lines serve the dual purpose of specifying an interface and offering a convenient way to automatically issue *Anetd* commands.

GUI lines have the following format:

```
<Anetd_command>&[submit!<submit_label>]&[new]&[add]&<any>{!|?}<port>&<any>{?|!}<host>[&any{!|?}<arg>]+
```

where:

- $< Anetd\_command >$ is one of the possible *Anetd* commands (LOAD, QUERY, INIT, CONF, GET, PUT, KILL, GETACL, GETWEB) to be invoked. This field determines which *Anetd* command will be issued upon activation of this GUI line.
- The optional *submit*! $< submit\_label >$ argument pair specifies that a clickable button with the label $< submit\_label >$ should appear to the user. When this button is pressed, one or more *Anetd* commands are executed.
- The optional *new* keyword specifies that, when the submit button is pressed, a new window is created by the GUI to display the results. If *new* is missing, results overwrite the current window.
- The optional *add* keyword specifies that this command should be automatically executed after the execution of the command specified in the previous GUI line.
- !|? separates an argument tag from its default value. ! specifies that the argument should be hidden to the user, and ? means that a value should appear in an input field and should be editable by the user. When the argument is editable, the tag is used to describe the input field.
- $< any >$!|? $< port >$ is a required argument pair that directs the *Anetd* command to an appropriate port.
- $< any >$?|! $< host >$ is a required argument pair that directs the *Anetd* command to an appropriate host.
- [*any*!|? $< arg >$]+ is a list of tag/value pairs that are used to complete the specification of the command. $< any >$ is any character string.

When GUI lines are returned, they are interpreted by a local client's script to generate HTML forms. GUI lines of the *text/ancors* MIME can therefore indirectly also issue any *Anetd* command. For example, the following GUI line

GET&new&submit!Get&port!3322&hostname?grumpy.csl.sri.com&file!file.out

would automatically produce the HTML form of figure 3

```
<FORM ACTION=file:exec.ax method=get TARGET=new>
<INPUT TYPE=SUBMIT VALUE=LOAD>
<INPUT NAME=exec VALUE=LOAD TYPE=HIDDEN>
<INPUT NAME=port VALUE=3322 TYPE=HIDDEN>
<INPUT NAME=host VALUE=grumpy.csl.sri.com TYPE=text>
<INPUT NAME=file VALUE="file.out" TYPE=text>
</FORM>
```

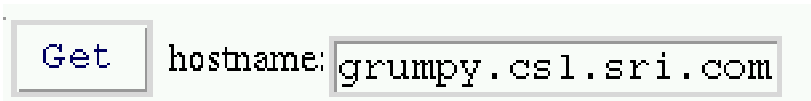

Fig. 3. GUI Example

This form would not be correctly interpreted and executed by a standard HTML browser, and it is therefore necessary to process it with the ANCORS front end. Notice that the action is the local file *exec.ax*; this local script is responsible for parsing the form arguments and issuing the LOAD command. When the user presses the clickable button, the *Anetd* command

```
GET 3322 grumpy.csl.sri.com file.out
```

is executed, and the returning output is displayed in a newly created window.

The ancors GUI front end was developed specifically to facilitate the use of *Anetd* and its related deployment capabilities and is fully integrated with the ANCORS executable specifications to properly handle *text/ancors*, simple *text/text*, and standard *text/html* MIME types. The ANCORS GUI front end was derived from the public-domain HTML browser called "Plume" written in TCL [6]. The browser was modified to invoke local scripts as well as remote scripts through the CGI protocol. The local scripts denoted with the extension *ax* are responsible for invoking various *Anetd* commands triggered by the clickable buttons.

5.2 SNMPV2 MIME Type

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 specifies the URL of the SNMP agent, the associated MIB, and access control information to be loaded with the service. It causes the management station to use *Anetd* to load the requested SNMP agent and MIB and to either spawn an SNMP browser to access the deployed service or update an existing SNMP management application.

5.3 Java_nm MIME Type

This example shows how CMF can be used to implement active NM mechanisms. In this approach the service replies by providing Java applications or applets that can then be embedded either in the central management stations or in a distributed monitoring agent, perhaps in a scheme similar to the one proposed in [2]. A reply using this MIME type might be something like

```
content-type application/java_nm

<JAVA>
<AGENT APPLICATION=http://www.abone.org/java/agent PARAM=<parameters>>  or
<AGENT APPLET=http://www.abone.org/java/agent_appl PARAM=<parameters>>
<AGENT FILES=http://www.abone.org/java/mib.txt FILES=http://www.activate.org/java/acl>
<AGENT_ENVIRONMENT VAR=MIBFILE VAL=mib.txt>

<MANAGER APPLICATION=http://www.abone.org/java/manager PARAM=<parameters>>  or
<MANAGER APPLET=http://www.abone.org/java/manager_appl PARAM=<parameters>>
<MANAGER FILES=http://www.abone.org/java/mib.txt FILES=http://www.activate.org/java/acl>
<MANAGER_ENVIRONMENT VAR=MIBFILE VAL=mib.txt>
</JAVA>
```

Notice that the MIME can either specify an application or an applet for both the agent side and the manager side. The application would not require special support other than an interprocess communication mechanism with the EE and the NM station, respectively. In this mode one only exploits the portability of Java to implement ad-hoc management protocols, and the solution may not be functionally different from what is provided by SNMP. With the specification of an applet, one can instead provide a mechanism that exploits active networking concepts. In this case the applet is assimilated by either a management or an agent EE, thus effectively extending the NM system at runtime. To support this latter mode of operation, it is possible to reuse existing active network software prototypes (such as ANTS [13]).

5.4 Active Management Information Base MIME

This example shoes how KBM can be used within the CMF design. KBM focuses on the fact that the data formats of standard NM frameworks (SNMP, CMIP) do not provide semantic interpretation of the MIB data. This limits their applicability to static and ad-hoc management where human intervention is necessary to interpret the meaning of specific management information and manually configure the management infrastructure. In dynamically changing networks, it will be necessary to (1) dynamically extend MIBs to include new management entities that come into existence and (2) automate the task of recognizing the semantic meaning of new or modified management primitives, to avoid frequent and costly human management operations.

In our design, an active network MIB is defined using multiple dimensions. The first dimension is analogous to the current MIB specification in which management information is organized in a tree structure reflecting the logical decomposition of the information. Figure 4 details how we plan to initially hierarchically organize active object definitions. This structure is deliberately very general because it is intended to be extended at runtime with new kinds of objects that are brought into existence with the deployment of new network services.

Active NM objects will also be defined by a vector of attributes, each of which is chosen from a discrete set of values and characterizes the objects in other dimensions. As an example, the following attributes may introduce useful additional dimensionality.

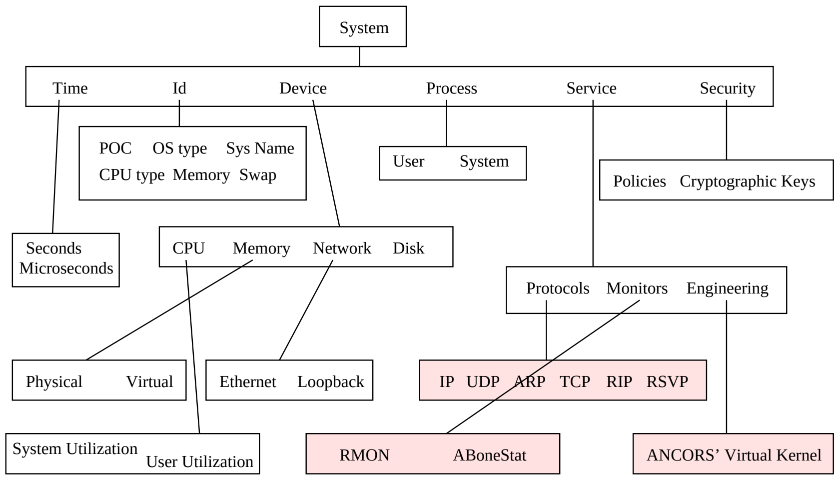

Fig. 4. Example Top-level MIB definition

- Criticality. This attribute specifies to what extent the value of the object is critical for the correct operation of the system. For example, (0) would mean that the value of the object does not affect operations, (1) would mean that the value of the object has significance at the local level, and (2) would mean 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) would mean that the value(s) of the object does not change, (1) would mean that the value(s) of the object changes with the frequency of human interactions (every seconds to minutes), and (2) would mean 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) would mean that the object is public and anyone can access it, (1) would mean that the object is accessible by authorized users and that access control should be enforced, and (2) would mean 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 0,1,0 to indicate that this object has a noncritical 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 it should be accessed only by privileged users.

A deployed service could cause the extension of an active MIB in an NM station as follows:

```
content-type application/anetm

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

The above reply might be returned by an activated RSVP service such as ARP [3]. This reply exports management operations and indicates that these operations should be used to extend a base active MIB 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. Analogously executing the **INIT** probe on other EEs may further extend the local active MIB with other functions particular to that EE[4].

This NM paradigm requires the development of a special-purpose NM application capable of correctly interpreting the information returned by the service. This application, after querying the EE with the **INIT** probe, will automatically add the remote procedure stubs in appropriate threads. The threads in the NM application will in turn reflect the values of the attributes exported by the deployed service, thus providing different types of polling modalities, access control mechanisms, and priority. Each time a remote procedure is called (whether the initial **INIT** call or subsequent remote procedure calls), it will respond with a MIME-encapsulated reply that will be recursively interpreted by the NM application. Leaf calls (procedures that only return data objects) may replay either with simple text MIMEs (which will simply display textual information), HTML formatted text, or special-purpose MIMEs designed to decode a particular binary encoding. The remote procedure stubs will also be classified, in the traditional manner, according to their structural role. They will be grouped into tree structures to facilitate the observation and control of the services from a process-oriented perspective.

[4] Collisions may occur when two different services choose the same name for an object. The colliding objects will be renamed using appropriate informative formats.

6 Conclusion

The dynamism of adaptable networks will require novel management and assessment techniques and tools to aid in the design, deployment and operation of network services. The integration of these different management aspects in a unified paradigm such as the one proposed in ANCORS requires a very flexible and open management framework that can be used in different ways. We have proposed a simple, open mechanism to support a plurality of management paradigms while observing a consistent architectural view. We have also outlined four different management approaches, from simple HTML based to more complex knowledge based, that show the generality of our open management framework and that will be used as case studies in the management of the ABONE.

References

1. D. Scott Alexander, Marianne Shaw, Scott M. Nettles, and Jonathan M. Smith. Active bridging. *Proceedings of the ACM SIGCOMM'97 Conference*, Cannes, France, September 1997.
2. F. Barillaud, L. Deri, and M. Feredun. Network management using internet technologies. *Integrated Network Management V*, San Diego, 1997.
3. R. Braden. Active signaling: the arp project. *In OPENSIG Workshop, University of Toronto*, Toronto, CA, October 1998.
4. J. Hartman, U. Manber, L. Peterson, and T. Proebsting. Liquid software: A new paradigm for networked systems. Technical Report 96-11, University of Arizona, 1996.
5. U. Legedza, D. J. Wetherall, and J. V. Guttag. Improving the performance of distributed applications using active networks. *Submitted to IEEE INFOCOM'98*, 1998.
6. J. K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.
7. P.A. Porras and P.G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. *Proceedings of the National Information Systems Security Conference*, Baltimore, October 1997.
8. Livio Ricciulli. ABONE: Active network back bone. *http://www.csl.sri.com/ancors/abone*, 1998.
9. Livio Ricciulli. ANETD: Active NETwork Daemon. Technical report, Computer Science Laboratory, SRI International, 1998.
10. Livio Ricciulli. High-fidelity distributed simulation of local area networks. *Proceedings of the 31st Annual Simulation Symposium*, Boston, April 1998.
11. Livio Ricciulli and Phillip A. Porras. ANCORS: An adaptable network control and reporting system. *Integrated Network Management V*, Boston, 1999.
12. Jonathan Smith, David Farber, Carl A. Gunter, Scott Nettle, Mark Segal, William D. Sincoskie, David Feldmeier, and Scott Alexander. Switchware: Towards a 21st century network infrastructure. *http://www.cis.upenn.edu/ switchware/papers/sware.ps*, 1997.
13. D. J. Wetherall, J. V. Guttag, and D. L. Tennenhouse. ANTS: A toolkit for building and dynamically deploying network protocols. *Submitted to IEEE OPENARCH'98*, 1998.

14. Y. Yemini and S. da Silva. Towards programmable networks. *IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, L'Aquila, Italy, October 1996.