An Adaptable Network COntrol and Reporting System (ANCORS) †

Livio Ricciulli
livio@csl.sri.com

Phillip A. Porras
porras@csl.sri.com

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

1 July 1998

Abstract

We present ANCORS, an adaptable network control and reporting system that merges technology from network management and distributed simulation to provide a unified paradigm for assessing, controlling, and designing active networks. ANCORS introduces a framework to assist in managing the substantial complexities of software reuse and scalability in active network environments. Specifically, ANCORS provides an extensible approach to the dynamic integration, management, and runtime assessment of various network protocols in live network operations. We present some of the advantages that can be obtained by merging technology from network management, distributed simulation, and active networking, and describe how ANCORS leverages complementary elements of each. We also introduce an ANCORS facility called the active network daemon *anetd*, which supports the deployment and system management of a large class of legacy software and newer active network applications under the ANCORS framework. Lastly, we present some prototype network engineering and monitoring services that were developed to demonstrate ANCORS's capabilities.

*† 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.

An Adaptable Network COntrol and Reporting System (ANCORS)

Abstract

We present ANCORS, an adaptable network control and reporting system that merges technology from network management and distributed simulation to provide a unified paradigm for assessing, controlling, and designing active networks. ANCORS introduces a framework to assist in managing the substantial complexities of software reuse and scalability in active network environments. Specifically, ANCORS provides an extensible approach to the dynamic integration, management, and runtime assessment of various network protocols in live network operations. We present some of the advantages that can be obtained by merging technology from network management, distributed simulation, and active networking, and describe how ANCORS leverages complementary elements of each. We also introduce an ANCORS facility called the active network daemon *anetd*, which supports the deployment and system management of a large class of legacy software and newer active network applications under the ANCORS framework. Lastly, we present some prototype network engineering and monitoring services that were developed to demonstrate ANCORS's capabilities.

1 Introduction

The most significant trends in network architecture design are being driven by the emerging needs for global mobility, virtual networking, and active network technology. The key property common to all these efforts is *adaptability*: adaptability to redeploy network assets, to rewrite communication rules, and to make dynamic insertion of new network services[1] a natural element in network operations. Critical to the deployment and management of these future networks is the need to provide consistency and control over dynamic changes, and to limit the impact that such changes have on performance and stability, as required for robust communication. Adaptive computing environments could benefit greatly from several ongoing research efforts.

Active network research [1, 11, 15, 25, 29, 30], in particular, seeks to pursue this concept of adaptive computing by providing network protocols that are more flexible and extensible. 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. Active networking tries to accommodate changes to network software by facilitating the safe and efficient dynamic reconfiguration of the network. Adaptive computing environments may be seen as the composition of the two main orthogonal approaches to active network design discussed in [29]:

- 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.

[1] In this paper, the term *network service* refers to a resource made available through the network that provides a well-defined interface for its utilization.

- 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.

In addition to work in the active network community, new standards are being proposed to assist in the distribution and maintenance of end-user applications [26, 27]. These standards attempt to introduce more timely and cost-effective mechanisms for distributing and maintaining application software via the network, allowing users to install or update software components by simply accessing HTML-like pages. However, extending such mechanisms to include the deployment and maintenance of system-level software is more difficult. The addition of system-level networking software must be done carefully to avoid potentially costly mistakes, and must also be properly coordinated with the management infrastructure if such changes are to be properly monitored and controlled.

While the trend toward adaptable protocol and application-layer technologies continues, the control and assessment of such mechanisms leaves open broader questions. Future networks could greatly benefit from simulation services that would allow network engineers to experiment with new network technologies on live network operations, without compromising service. Live-traffic-based simulation services would provide engineers insight into how a proposed alteration would affect a network, without committing the network to potentially disruptive consequences.

Finally, the management of adaptive networks would greatly benefit from sophisticated monitoring tools to help assess the effects of runtime alterations and detect when those effects result in activity outside a boundary of desired operation. Current network management (NM) and control software is oriented toward servicing single administrative domains. As new interdependencies arise in sharing resources beyond single administrative domains, monitoring capabilities (e.g., performance monitors, fault- or misuse-detection services) should be able to change over time, adapt as new conditions develop, and scale well to large networks.

ANCORS is intended to streamline and, at the same time, enrich the management and monitoring of active networks, while adding new support to the network management paradigm to assist network designers. Section 2 presents the core benefits that ANCORS provides by merging active networking with network management and distributed simulation. Section 3 discusses ANCORS's flexible system management framework, which centers around the introduction of an ANCORS active network daemon (*anetd*). The *anetd* facility allows ANCORS to manage the dynamic deployment and control of new services within a network. Section 4 describes our initial experiences in implementing new simulation and monitoring services under ANCORS, and Section 5 discusses related work.

2 ANCORS Paradigm

The ANCORS project is pursuing a unified paradigm for managing change in active network computing environments. Underlying this framework is a conceptual model for how elements of technology from network management, distributed simulation, and active network research can be combined under a single integrated environment. This conceptual model is illustrated in Figure 1.

ANCORS gains from *discrete* active networking the ability to dynamically deploy engineering, management, and data transport services at runtime. ANCORS leverages this capability with network management technology to (1) integrate network and system management with legacy standards (SNMP, CMIP) to provide a more flexible and scalable management framework, (2) provide network management functions to

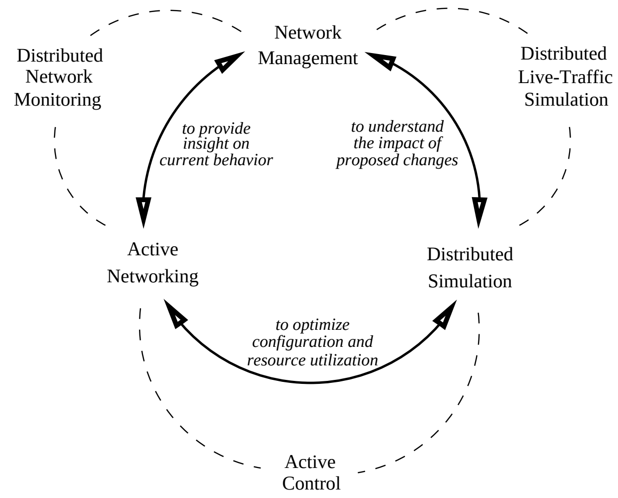

Figure 1: A Synergistic Model for Management, Simulation, and Active Control

supervise network operations, (3) dynamically deploy mechanisms to collect network statistics to be used as input to network engineering tools and higher-level assessment tools, and (4) assist network operators in reacting to significant changes in the network. In addition, ANCORS integrates distributed simulation facilities that will observe live traffic, and simulate/emulate the effects that proposed modifications have on the network.

2.1 Active Node Architecture

The coexistence of the discrete and integrated active networking approaches increases network adaptability. Therefore, besides offering flexible integrated execution environments (EEs), such as those being developed by the active network community, adaptable networks should also support the coexistence of concurrent programming environments that can be discretely deployed within a single node.

This has prompted the formulation of the active network node architecture [10] depicted in Figure 2. Here, several EEs execute in user space and explicitly request resources from the node operating system. A particularly interesting aspect of the architecture is its provision for a privileged management EE that performs network and system management tasks (security implications are discussed in Section 3.1.3). In addition, a demultiplexer dispatches active network packets encapsulated through ANEP (Active Network Encapsulation Protocol) to the different EEs (including the developing new management and engineering technology that fits well within this design (see Section 4).

2.2 ANCORS Architecture

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

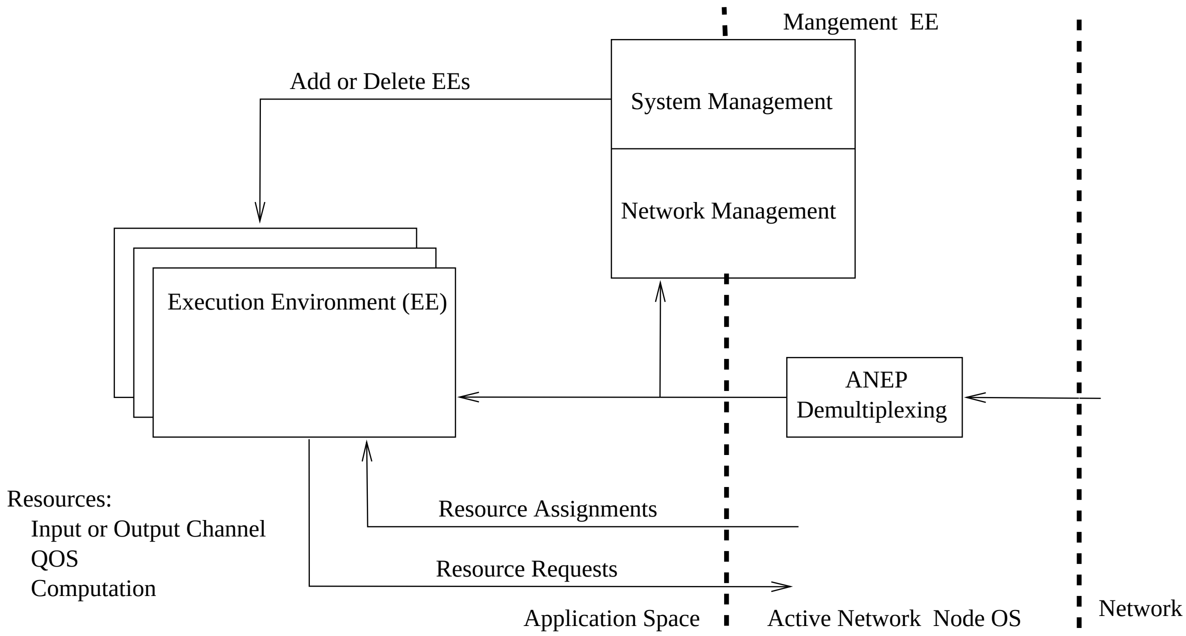

Figure 2: Active Network Node Architecture

architecture map very well to already existing technology. Recognizing this, the architecture has been explicitly designed to accommodate other network management engineering solutions.

The ANCORS architecture is divided into data, assessment, and control layers, as illustrated in Figure 3. 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 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, [2] 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.[3] The following sections describe these logical layers in the ANCORS architecture.

2.2.1 Data Layer

The foundation of the ANCORS architecture is the data layer, which is composed of engineering, monitoring, and data transport services. Although presented as a single layer, it is useful to recognize and distinguish the various modules that may populate this layer. For this reason, we decompose the data layer

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

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

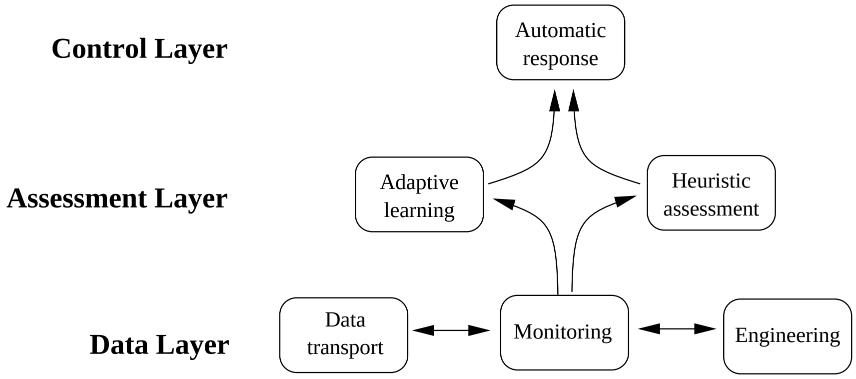

Figure 3: The ANCORS Architecture

into three distinct data service types, all of which may benefit from dynamic deployment in the network.

Data Transport Services Data transport services offer communication protocols that are either quite general and extensible as proposed in [1, 11, 25, 29, 30], or that support more traditional services derived from those available today. Although data transport services are an integral part of the ANCORS architecture, the project does not focus on the development of new transports. Rather, the architecture is tailored to accommodate both new and legacy data transport services. In all cases, we assume that data transport services may be dynamically deployed. The scope of this paper does not permit us to give a detailed description of these services and their possible applications. We instead focus on the monitoring and engineering services.

Monitoring Monitoring services perform tasks analogous to those of today's network management agents. In general monitoring services record the operation of other network services, perform analytical reviews of the network traffic (either directly or through the review of data collected by legacy monitoring agents) and can report relevant information to higher-layer components in the ANCORS architecture. A variety of legacy SNMP- or CMIP-based agents like RMON may be directly integrated into the ANCORS framework. In addition, specialized network monitoring services may be dynamically deployed to perform user-defined targeted analyses, such as those discussed in [17].

The use of active networking to dispatch user-definable monitoring capabilities gives ANCORS two major advantages: (1) it permits selective monitoring of a particular phenomena, such as new network requirements and new usage patterns that emerge over time, and (2) it improves monitoring scalability (as suggested in [16]) through various degrees of sophistication in the monitoring agents, thus allowing, as suggested in [13], a fluid tradeoff in the amount of computation performed in the services distributed throughout the network and the amount of computation performed at the management station (see Section 4.2). These services are deployable in the NM EE of the active network node discussed in Section 2.1.

Engineering Services Engineering services aid in the design and testing of network services before their deployment. ANCORS incorporates distributed simulation to help operators explore and select the optimal deployment and configuration of network assets.

ANCORS's engineering support leverages network management and introduces simulation as an additional network service. Integrating distributed simulation in the ANCORS paradigm has four main advantages: (1) it naturally supports reuse of both simulation software and network models because the models can be remotely accessed as any other software component, (2) the simulated models can naturally use real network data produced by the monitoring agents, thus improving fidelity, (3) the consumers of the data (the simulation models) are placed close to the origin of the data to reduce overhead, and (4) the monitoring capabilities of network management can be reused to monitor and control the simulations.

In practice, engineering services may mimic the behavior and performance of all other network services, but differ in the following ways: (1) they live in a separate address space and are for the exclusive use of the network designers, (2) they execute protocols in a virtual timescale that may differ from physical time,[4] and (3) they generate synthetic network traffic that does not contain user data. (See section 4 for an example network simulation service.)

2.2.2 Assessment Layer

The ANCORS architecture has been designed to be able to reuse and integrate software components derived from significant advances in network alarm correlation, fault identification and distributed intrusion detection. In particular, this layer of the ANCORS architecture performs tasks analogous to alarm correlation and fault analysis of the types currently proposed by network management expert systems [6, 14, 24, 28]. Such analyses may be performed using two different approaches. The first, and more common approach, is a heuristic review of elements within the data, specifically looking for metrics within the reported results that represent exceptional or unexpected behavior. Boundary results for the metrics are specified at the initialization of the data collection agents, and may be dynamically updated by an administrator. The second form of assessment involves adaptive learning, where continuous statistical profiling of the network data provides dynamically updated metrics from which unusual conditions may be detected. The algorithms used to provide the heuristic assessment and the statistical profiling of network data may be adapted from related research on information survivability [18].

2.2.3 Control layer

The output from the assessment layer is propagated to the control layer, where results are displayed for the administrator or passed to a decision engine capable of invoking predefined target-specific response methods. Included with each valid response method are evaluation metrics for determining the circumstances under which the method should be dispatched. Formulating effective responses to detected exceptional activity is, itself, an extensive subtopic being pursued within the scope of our research. In many situations, the most effective response may be no response at all, in that every response imposes some cost in system performance or (worse) human time. It is important to tailor a response that is appropriate given the severity of the problem, and that provides a singular effect to address the problem without harming the flow of legitimate network traffic. We are broadly pursuing four types of control-layer responses:

- **Passive results dissemination:** Analysis results are made available for administrative review. Already-existing network protocols such as SNMP, including the translation of analysis results into an ad-hoc MIB structure, are examples of passive reporting.
- **Assertive results dissemination:** Notifications or alerts (e.g., via email, console alert or paging) may be sent directly to the administrator for more timely recognition.

[4] The passage of time is explicitly controlled by predefined time-synchronization algorithms.

• **Dynamic oversight of the monitoring layer:** Configuration changes may be provided directly to the monitoring facilities of the data layer (i.e., a feedback loop is established between control and monitoring).
• **Dynamic configuration changes:** Changes can be made directly to the active network configuration in response to exceptional activity. Such actions may include reconfiguration of services, routing databases, or filtering rules.

3 ANETD: Active NETworks Daemon

Anetd is ANCORS's facility for managing the deployment, operation, and monitoring of deployable network services in an active network. *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 facilitate the porting of existing software. However, *anetd* is not intended to deploy a large number of libraries or require large installation directories.[5]

From a system management perspective, *anetd* views all services within the ANCORS architecture (Figure 3) 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 is illustrated in Figure 4.

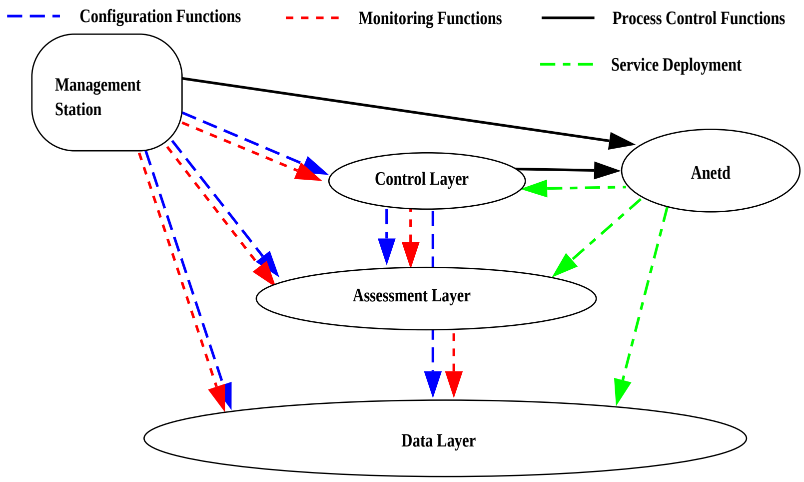

Figure 4: ANCORS's Management Architecture

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 4 that ANCORS allows the traditional but nonscalable approach of having the management station directly monitor and control the data layer. This aspect of our architecture can

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

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).

3.1 *Anetd* Prototype

Anetd listens on a user-assigned UDP port and accepts ANEP encapsulated packets. If the type ID in the ANEP packet header is of the type assigned to ANCORS by the Internet Assigned Numbers Authority (IANA) (type 51), the ANEP payload is parsed and the *anetd* control commands described in Section 3.1.1 are interpreted. A client application encapsulates ANCORS system management commands using an ANEP header, and forwards the commands to *anetd*. In addition to processing commands directed to itself, *anetd* looks at the packets it receives and, if a packet is destined to one of the services that was previously deployed, it forwards that packet to the appropriate process. This is accomplished as follows: once a new application with an ANEP type assigned to it is deployed, *anetd* adds the type ID to a demultiplexing table. The demultiplexing table maps an ANEP type ID to a file descriptor; the file descriptor, in turn, maps to the standard input of the application. Application output streams do not require any support from *anetd* and can allow the system to assign output ports automatically.[6]

3.1.1 System Management Commands

Anetd control commands allow a client to deploy, configure, and manage network application software. Our current prototype supports the deployment and control of either UNIX binaries or Java applications.[7]

The LOAD command instructs *anetd* to download files specified via URL and to start a new network service. The load command has the format:

LOAD [T=< anepid >] [J=< jurl > | X =< url > | A =< url >] [F=< url > ...] [S=< val > ...] [E=< var : val > ...] [D=< dir >] [O=< file >] [R=< file >]. Each active network EE has a unique ANEP ID, which is administratively assigned by IANA; *Anepid* is the ANEP ID of the service being deployed. If this argument is not present, the deployed service will be assigned type 0 and no packets will be demultiplexed to it by *anetd*. *J=< jurl >* specifies a Java application. *jurl* is of the form

```
http://servername.edu:port/classpath/~class
```

where *servername.edu:port* specifies the server where the Java application is located following normal URL conventions. The *classpath* is a path pointing to the base classpath of the Java application. The *class* is the class to invoke. In this type of deployment, *anetd* simply triggers Java's built-in mechanisms to dynamically resolve the methods required by the application throughout the base classpath information. After proper initialization, Java's Virtual Machine automatically downloads the required bytecode from the specified HTTP server, following the RMI specification [12].

X= url specifies the URL of a binary-compatible file *url*. After the file is downloaded with an HTTP `GET` command and the HTTP header is stripped, the downloaded code is simply forked as a separate child process.

[6] We believe that this demultiplexing scheme is analogous to the mechanism that is used in *inetd*.

[7] The currently supported UNIX platforms are Linux 2.0, SunOS 5.5 and FreeBSD 2.2.

$A=$ *url* indicates that *url* specifies an ANCORS image. An ANCORS image is compiled as a dynamic library, and it contains application-specific code to be added to a remote process. After downloading the ANCORS image in an analogous way to the code for the X type, *anetd* opens the image as a dynamic library and invokes the symbol *init_model*. *Anetd* then returns the handle for the new image, together with any strings returned by the *init_model* function, to the client that issued the load command. The client can then later refer to the specific handles returned by *anetd* to invoke application-specific procedures through the CONF command. ANCORS images can therefore be used as building blocks to compose separate native-code services coexisting in a single process.

Because the X and A types specify native executables, *anetd* automatically appends an extension to the specified URLs that matches the particular operating system on which *anetd* is running. Other flags (S,E,D,O,R) specify execution parameters and the flag F specifies URLs of data files to simply upload through *anetd*. Space restrictions do not allow us to describe these flags; please refer to [20] for a detailed description.

The CONF Command applies to ANCORS images and is similar to a remote procedure call. The CONF command has the format *CONF < handle > < symbol > [args ...]*, where *handle* specifies the image to which the CONF command applies, *symbol* specifies the name of the function to invoke, and *args* are a sequence of command line arguments to be passed to the function *symbol*.

The QUERY command returns a list of network services that were forked by *anetd* to the client originating the command. The QUERY command format is simply *QUERY*. The list of forked services has the format *< index > < clientip > < description >*, where *index* is an index generated by *anetd* (0,1,2,...), *clientip* is the IP address of the client that installed the service, and *description* is a textual description of the service.

The KILL command terminates a network service. The KILL command has the format *KILL < index >*, where *index* is the index of the thread to be terminated by *anetd* as provided by *QUERY*. *Anetd* automatically garbage-collects all resources allocated to the terminated service and only allows the client that originally deployed the service to perform the operation.

3.1.2 Deploying ANTS

We include an example demonstrating how a client can deploy ANTS [29] active nodes. This example uses a program named *sc*, which simply encapsulates ANCORS *anetd* requests with ANEP. Three different ANTS active nodes can be deployed on host1, host2, and host3 and pipe their standard output to log files by using the following commands:

```
sc 3322 host1 LOAD
J=http://sequoia.csl.sri.com:7000/ants-1.2.a/~ants/ConfigurationManager
F=http://sequoia.csl.sri.com:7000/ants-1.2.a/runs/data.config
F=http://sequoia.csl.sri.com:7000/ants-1.2.a/runs/data.routes
S=data.config S=18.31.12.3 T=18 O=log
sc 3322 host2 LOAD
J=http://sequoia.csl.sri.com:7000/ants-1.2.a/~ants/ConfigurationManager
F=http://sequoia.csl.sri.com:7000/ants-1.2.a/runs/data.config
F=http://sequoia.csl.sri.com:7000/ants-1.2.a/runs/data.routes
S=data.config S=18.31.12.2 T=18 O=log
sc 3322 host3 LOAD
```

```
J=http://sequoia.csl.sri.com:7000/ants-1.2.a/~ants/ConfigurationManager
F=http://sequoia.csl.sri.com:7000/ants-1.2.a/runs/data.config
F=http://sequoia.csl.sri.com:7000/ants-1.2.a/runs/data.routes
S=data.config S=18.31.12.1 T=18 O=log
```

These commands are sent to three *anetd* daemons, located on host1, host2, and host3, which listen on port 3322 (the active network port). The LOAD commands cause the three *anetd* daemons to instantiate copies of the ANTS active network nodes and pipe their standard output to some log files. Notice that the ANTS application specified by *J=< jurl >* requires two configuration files loaded by the *F=< url >* arguments, and the command-line arguments *S=data.config S=18.31.12.*{*1,2,3*}. The *T=18* argument instructs *anetd* to later forward ANEP packets of type ID 18 to this application. To illustrate the generality of *Anetd*, we also give an example of how a native code SNMP agent could be deployed using *anetd*:

```
sc 3322 host1 LOAD
X=http://sequoia.csl.sri.com:7000/SNMP/snmpd
F=http://sequoia.csl.sri.com:7000/SNMP/mib.txt
F=http://sequoia.csl.sri.com:7000/SNMP/acl.conf
F=http://sequoia.csl.sri.com:7000/SNMP/party.conf
F=http://sequoia.csl.sri.com:7000/SNMP/view.conf
F=http://sequoia.csl.sri.com:7000/SNMP/context.conf
S=-p
S=8000
E=MIBFILE:mib.txt
```

An SNMP agent called *snmpd* is deployed to host1 with all the appropriate configuration files and command-line arguments. Note that the *T* argument is missing, signifying that *anetd* is not required to demultiplex ANEP packets to the SNMP daemon.

3.1.3 Access Control

Network security is a very important issue that should be addressed during all stages of design and implementation of any network software. *Anetd* offers two ways for providing access control: (1) it only executes deployment and control commands originating from a set of known IP addresses, and (2) it only accepts code originating from a set of known HTTP servers. The access control information is specified in two files residing on the server: *host.allow* and *web.allow*. *Hosts.allow* contains the list of hosts authorized to send control commands to *anetd*. *Web.allow* contains a list of HTTP code servers from which *anetd* is authorized to retrieve code. Accepting control commands only from a set of clients allows administrative authority to be set, thus limiting misuse of the daemon. Restricting code retrieval from a set of secure code servers will help limit the potential for misuse of ANCORS system management service.

As with most other networking applications, *anetd*, for the time being, is not secured by cryptographic means, and therefore is open to spoofing attacks. The security model for the active network node architecture is currently being defined. As this security framework is finalized, *anetd* will integrate its access control policies with cryptographic services provided by the active network node. Multiple *anetd* daemons can be running simultaneously on the same host, perhaps implementing multiple virtual active networks. The only restrictions are that (1) multiple daemons cannot share the same ANEP port, and (2) if multiple daemons are controlled by the same client, care must be taken in avoiding overwriting configuration files and output redirection files.

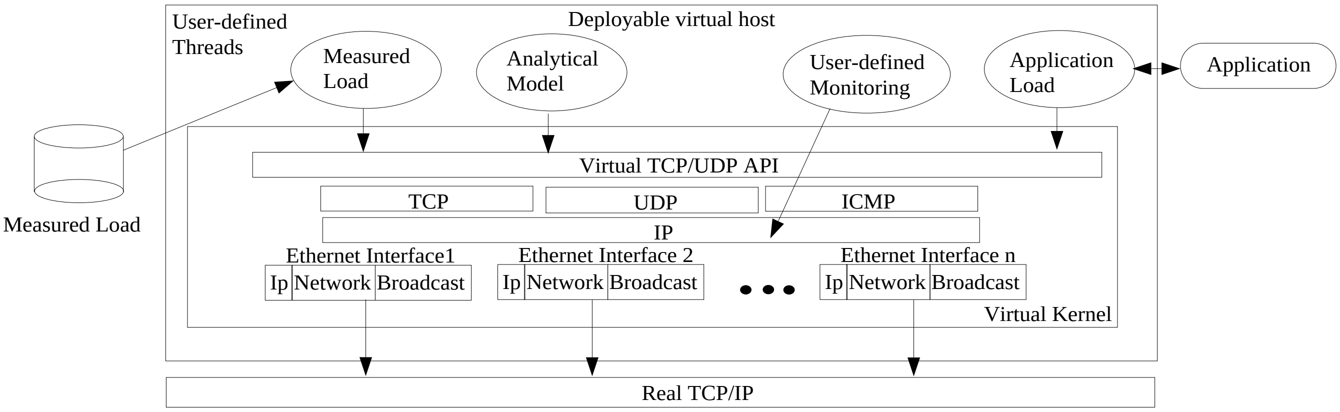

Figure 5: Deployable Virtual Host

4 Virtual Networking Using ANCORS

Anetd has been designed to support the deployment and management of a large class of active network applications. In addition to this system-level support, *anetd* offers specialized functionality to ANCORS native-code applications. These applications can be deployed and managed as any other third-party active network application, but they also exploit a set of built-in primitives that, in addition to standard native system functionality (I/O, memory management, networking), provide (1) multithreading (nonpreemptive), (2) reliable multicast emulation, and (3) global time synchronization. These primitives were designed to provide support for distributed simulation network engineering applications, as well as some forms of sophisticated network monitoring. A detailed description of these services is presented in [19], and briefly summarized here.

To date, we have produced a representative example of an engineering network service that emulates a UNIX kernel. This service, depicted in Figure 5, was obtained by modifying a Linux operating system to allow its execution as a user process. The modifications of the operating system substituted the lower-level hardware-dependent procedures and interfaces with user-level counterparts. We deleted the file system support and incorporated all necessary configuration procedures (like *ifconfig* and *route*) into the virtual kernel itself. Memory management was completely deleted and replaced by user-level memory allocation functions (*malloc* and *free*). The scheduling was also completely replaced by nonpreemptive threading offered by the simulation package CSIM [23].

The resulting service executes on a virtual timescale, and offers the identical networking behavior of a real Linux kernel, providing a vehicle to instantiate high-fidelity distributed simulations of virtual networks [19]. One of the model's configuration functions accepts several different timing configurations to approximate the protocol stack timing behavior of four different kernels (SunOS 4.1.3, SunOS 5.5, Linux 2.02, and FreeBSD 2.2). The virtual kernel offers the network application programming interface (API) of the real Linux counterpart and therefore can be used to reproduce a wide range of loading conditions. ANCORS's ability to add and delete threads can be used in this application to dynamically change loading conditions (by adding or deleting user-defined loading threads) or by injecting user-defined monitoring probes into the kernel, so that specific parameters can be observed. As shown in Figure 5, the user-definable loads may be produced by either closely mimicking real load conditions recorded by network monitoring services or by linking some real applications to the virtual kernel to generate application-specific loads.

The deployment of a virtual network is achieved by downloading and configuring several virtual kernels through *anetd* daemons. All these operations can be performed either through a standard HTML browser or via script. We have so far instantiated several virtual networks running on a network of workstations including Sun SPARCstation 20s, UltraSPARCs, and Intel-based machines running FreeBSD and Linux. (In [19], we measured their performance and scalability.) We are currently using our prototyping system to evaluate several types of proposed defense mechanisms against the SYN flooding denial-of-service attack, as outlined next.

4.1 TCP SYN Defense: An Example of Adaptive Response

To illustrate how ANCORS can aid in improving the adaptability of current networks, consider a situation where a malicious entity attempts to deny access to an HTTP service via the TCP SYN flooding attack [7]. SYN flooding attacks are performed by the attacker submitting a continuous stream of TCP SYN (connection request) packets to the target system, thus reducing (or eliminating) the target system's ability to respond to legitimate connection requests.

In some situations, the inter-domain cooperation necessary to trace an attacker may not be possible. Thus, it is necessary to adopt some local defense mechanisms. Several backward compatible approaches have been proposed to addressing the SYN flooding attack, each of which has tradeoffs with respect to effectiveness, robustness, and resource requirements [21]. In particular, one solution greatly increases the allowed number of outstanding "connection-pending" entries, another does not allocate any connection entry until the very last message from the client, and yet a third requires five instead of three messages to open a connection. Unfortunately, under the first solution, unless limited by bandwidth, an attacker can easily circumvent the countermeasure by producing enough SYNs to fill the larger queue. The second solution breaks the TCP semantics in that the last packet may be lost because of congestion, thus creating a client in a frozen state. The third solution, although overcoming the shortfalls of the previous two, adds a significant overhead to each connection establishment.

We contend that the choice on which solution to adopt is intimately tied to the characteristics of the particular traffic patterns experienced by the server during the attack. For example, if the rate at which the attacker can send SYN is low enough with respect to the rate at which the target host services connection requests, then the first solution introduces the least cost. A target host with very low network congestion, but heavy SYN flooding, would prefer the second solution. However, a usage pattern that reflects relatively infrequent connection requests (more in the style of an FTP service) would prefer the third solution.

Figure 6 illustrates how ANCORS provides system management support to first record and characterize the HTTP traffic (possibly including a record of the SYN attacks) by deploying a user-defined monitoring service. The measured traffic is recorded in a format that can then later be used to reproduce the HTTP loading conditions in a synthetic environment. A modified TCP module that implements different SYN flooding defense mechanisms, and a new instantiation of the HTTP server are also deployed through ANCORS. Several simulated clients (with one being the attacker) are deployed to generate end-to-end virtual HTTP traffic.

Notice that the HTTP server and IP layer of the simulation correspond to the identical code used by the real system thus providing software reuse and high fidelity of the results while the TCP module is derived by modifying the original code. Several runs of the virtual network constructed through ANCORS can then be used by a network engineer to choose the best SYN flooding defense mechanism given current traffic. In addition, the simulation can also be used to forecast how the three mechanisms would behave given variations in the server traffic, the attacks, or both. Once a suitable TCP module is found, it can be directly inserted in the protocol stack of the real HTTP server without modification. We are currently prototyping some of these ideas, using ANCORS in a similar manner to quantitatively analyze SYN flooding

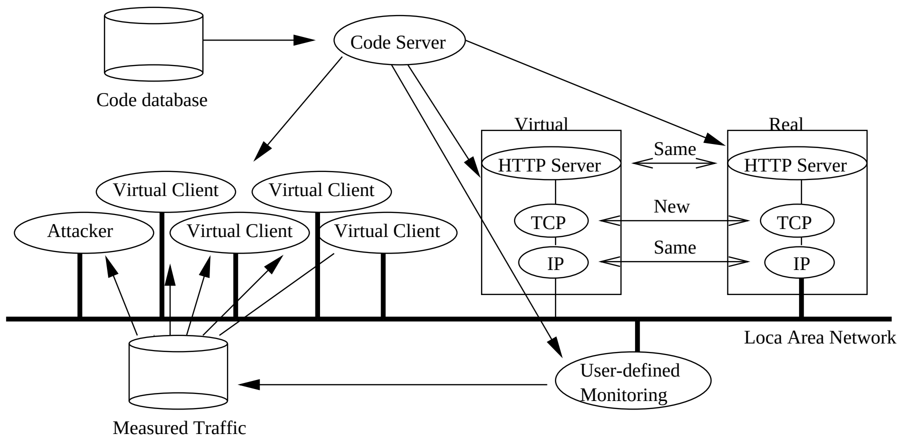

Figure 6: Use of Integrated Engineering Services

defense mechanisms, and to reliably predict their behavior in a controlled synthetic environment (see [21] for further details).

4.2 Dynamic Insertion of Monitoring Services

We are also developing a prototype downloadable intrusion-detection service for ANCORS, which represents a management by delegation monitoring service at the data layer. This service can be dynamically configured and deployed to any network element that hosts an ANCORS daemon, and can return analysis results to the assessment layer for correlation and perhaps control-layer responses. These dynamically distributable intrusion-detection service modules represent a significant departure from the previous centralized host-based, user-oriented, intrusion-detection work that suffer poor scalability and lack of integration into large networks. The intrusion-detection services are adaptations of separate intrusion-detection research tools developed under the EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) project [18], and represent one of the first distributed network surveillance security mechanisms that will fully integrate into an active network paradigm.

The intrusion-detection services consists of native event filters that interact with the analysis target to collect a variety of operational data, including audit data, system- or application-layer activity logs, or raw network traffic. The operational data is forwarded to two analysis engines, a signature analysis engine (i.e., an expert-system) and a statistical anomaly detection engine. These analysis modules are general-purpose and require instantiation with a pluggable configuration library, called a *resource object*, to tune them for a given analysis objective. The resource object is highly target-specific, containing all of the operating parameters necessary to customize the intrusion-detection modules for the target system, including the native methods needed to retrieve and format the target event stream. As intrusion-detection modules are deployed from one analysis target to another, no code modifications are necessary. Rather, one must construct, or select from a preconstructed library, the resource object appropriate for the target host and analysis objectives.

Figure 7 illustrates an example application, which we are currently developing using components inte-

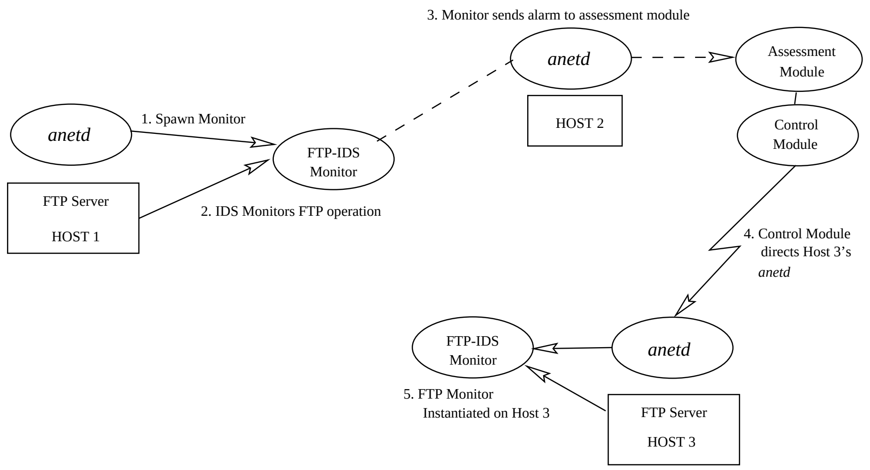

Figure 7: Intrusion Detection Service Deployment under ANCORS

grated from our adaptive network and intrusion detection research. The figure illustrates how intrusion-detection service modules can employ ANCORS to propagate alarm information to other hosts, causing them to dynamically load monitoring tools when a specific threat is encountered. In this scenario, an *anetd* operating on Host 1 spawns an FTP-specific intrusion-detection monitor, which analyzes network traffic exchanged between Host 1's FTP server and external clients. The FTP monitor converts the packets from the FTP control protocol (port 21 traffic) into an internal form, and inputs this data into a rule-based expert system and statistical anomaly detection unit. Both the expert system and anomaly detection unit are instantiated with resource objects tuned to detect exceptional FTP activity. In this scenario, the FTP monitor produces an alarm based on its analyses of FTP transactions, and forwards this alarm to an ANCORS assessment module located on Host 2. The assessment module then forwards the alarm to an ANCORS Control module, which in turn (via predefined response handling rules) sends a LOAD request to the Host 3 *anetd*. Lastly, the Host 3 *anetd* loads and spawns its own FTP intrusion detection service to search for activity similar to that encountered on Host 1. We have completed the deployment of the intrusion detection module, and are currently integrating the assessment and control modules.

5 Related Work

5.1 Code Mobility

The concept of being able to dynamically move management functionality from the management station out toward the managed object at runtime has long been proposed. Management by delegation [31] introduced this notion of moving management code on the network to where it is needed to decentralize management functions. The advent of the Web and later Java has resulted in a number of designs (see [3, 4, 8] for some examples) that exploit this powerful idea. More recent active networking ideas [5, 22, 25, 29] have extended the concept of management by delegation to allow the addition and modification of monitoring applications, as well as user-definable system-level network support that can directly interact with the communication resources.

ANCORS's support for dynamically downloading network services at runtime is different from previous work in that it places a much stronger emphasis on (1) bridging code mobility functionality with legacy systems and applications and (2) offering a generalized infrastructure for distributing and managing multiple kinds of distributed mobile applications. ANCORS offers an API for downloading and managing mobile code in a generic manner, using the Active Network Encapsulation Protocol [2] as its packet-level interface. ANCORS's system management support, therefore, is not targeted to a single language or application but can be used to bootstrap a large class of mobile application.

5.2 Engineering Support

A unique characteristic of the ANCORS architecture is that it integrates distributed simulation into the network management arena. To the best of our knowledge, no other work has been done on this subject. The closest work may that described in [9], where TCL scripts are downloaded in the network to decentralize the task of testing network behavior. Code movement and results retrieval are performed through SNMP to allow easier integration into standard NM tools. The most significant differences between ANCORS's capabilities and the cited work is that ANCORS directly supports distributed simulation rather than emulation, and it allows the deployment and control of models of arbitrary complexity. In fact, ANCORS models can be as complex as the active networking software itself.

6 Conclusion

As the dynamic deployment of network services becomes standard technology to support user applications, network operators will require an efficient and flexible infrastructure to assist them in network design, configuration, and monitoring. The quality of future network management, monitoring, and engineering tools and standards will be crucial in determining the speed at which networking will evolve toward a more dynamic architecture. In ANCORS, network monitoring, control, and design can coexist in an integrated paradigm. The synergy of combining distributed simulation, network monitoring, and active networking will dramatically increase the power of network management and engineering. We have shown how a unified, yet very extensible, system management framework can be derived from current Web technology to provide compatibility with legacy standard (SNMP) and virtually unlimited extensibility to introduce more powerful management technologies as they become available. We have also described services we have implemented that can be used in the context of our new network management framework.

In the near future, we will bridge our work with existing active networking technologies to provide an integrated platform for merging data transport protocols and their associated deployment mechanisms with our extensible engineering and management support. In addition, we will use our infrastructure to conduct network engineering experiments to advance the understanding of end-to-end network behavior and offer a user-friendly environment for the development of new network technologies.

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] Scott Alexander, Carl A. Gunter, Angelos D. Keromytis, Gary Minden, and David Wetherall. ANEP: Active network encapsulation protocol. *http://www.cis.upenn.edu/ switchware/ANEP/docs/ANEP.txt*, 1997.

[3] F. Barillaud, L. Deri, and M. Feredun. Customer facing components for network management systems. *Integrated Network Management V*, San Diego, 1997.

[4] F. Barillaud, L. Deri, and M. Feredun. Network management using internet technologies. *Integrated Network Management V*, San Diego, 1997.

[5] Samrat Bhattacharjee, Ken Calvert, and Ellen W. Zegura. An architecture for active networking. April 1997.

[6] A. Bouloutas, A. Calo, and A. Finkel. Alarm correlation and fault identification schemes in communication networks. Technical Report RC 17967, IBM, 1992.

[7] Computer Emergency Response Team (CERT). TCP SYN flooding and IP spoofing attacks. Technical Report CA-95:01, Carnegie Mellon University, September 1996.

[8] J.C. Gregoire. Models and support mechanisms for distributed management. In A. S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Integrated Network Management IV*, page 17. Chapman & Hall, 1995.

[9] G. Grimes and B. P. Adley. Intelligent agents for networking fault diagnosis and testing. *Integrated Network Management V*, San Diego, 1997.

[10] Active Networks Working Group. Architectural framework for active networks. Technical Report http://www.cc.gatech.edu/projects/canes/arch/arch-draft.ps, 1998.

[11] 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.

[12] Javasoft. The java remote method invocation (RMI) specification. Technical report, Javasoft, Sun Microsystems Incorporated. *http://www.javasoft.com/products/jdk/1.2/docs/guide/rmi/spec/rmiTOC.doc.html*, 1998.

[13] I. Katzela, A.T. Bouloutas, and S.B. Calo. Centralized vs. distributed fault localization. In A. S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Integrated Network Management IV*, pages 250–261. Chapman & Hall, 1995.

[14] I. Katzela, A.T. Bouloutas, and S.B. Calo. Centralized vs. distributed fault localization. In A. S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Integrated Network Management IV*, pages 250–261. Chapman & Hall, 1995.

[15] U. Legedza, D. J. Wetherall, and J. V. Guttag. Improving the performance of distributed applications using active networks. *Submitted to IEEE INFOCOM'98*, 1998.

[16] K. Meyer, M. Erlinger, C. Sunshine, G. Goldszmidt, and Y. Yemini. Decentralized control and intelligence in network management. In A. S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Integrated Network Management IV*, page 4. Chapman & Hall, 1995.

[17] P.A. Porras and A.Valdes. Live traffic analysis of TCP/IP gateways. *Proceedings of the Network and Distributed System Security Symposium*, San Diego, March 1998.

[18] 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.

[19] L. Ricciulli. High-fidelity distributed simulation of local area networks. *Proceedings of the 31st Annual Simulation Symposium*, Boston, April 1998.

[20] Livio Ricciulli. ANETD: Active NETwork Daemon. Technical report, Computer Science Laboratory, SRI International, 1998.

[21] Livio Riccuilli and Pankaj Kakkar. Defense against TCP SYN flooding. *submitted to the ISOC Symposium on Network and Distributed System Security*, 1999.

[22] Beverly Schwartz, Wenyi Zhou, Alden W. Jackson, W. Timothy Strayer, Dennis Rockwell, and Craig Part ridge. Smart packets for active networks. Technical Report http://www.net-tech.bbn.com/smtpkts/smtpkts-index.html, BBN Network Technologies, January 1998.

[23] H. Schwetman. CSIM: a C-based, process-oriented simulation language. Technical report, MCC, 1989.

[24] N. Shroff and M. Schwartz. Fault detection/identification in linear lightwave. Technical Report CU/CTR/TR 243-91-24, CTR, 1989.

[25] 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.

[26] Arthur van Hoff, John Giannandrea, Mark Hapner, Steve Carter, and Milo Medin. *The HTTP Distribution and Replication Protocol*. http://www.marimba.com/standards/drp.html, August 1997.

[27] Arthur van Hoff, Hadi Partovi, and Tom Thai. *Specification for the Open Software Description (OSD) Format*. http://www.microsoft.com/standards/osd/, August 1997.

[28] C. Wang and M. Schwartz. Identification of faulty links in dynamic-routed networks. *IEEE JSAC*, 11, 1993.

[29] 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.

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

[31] Y. Yemini, G. Goldszmidt, and S. Yemini. Network management by delegation. *Second International Symposium on Integrated Network Management*, Washington DC, April 1991.