An Adaptable Network COntrol and Reporting System (ANCORS)[*]

L. Ricciulli, P. Porras
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park,CA 94025
{livio, porras}@csl.sri.com

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. Last, we present a prototype network engineering service that was developed to demonstrate ANCORS's capabilities.

Keywords

Network engineering, active networks, distributed simulation, protocol prototyping, dynamic deployment

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 in-

[*] 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.

sertion 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 [2,4,7,15,19,20], 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 [19]:

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.

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.

2 ANCORS Paradigm

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 [16,17]. 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.

[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.

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. 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.

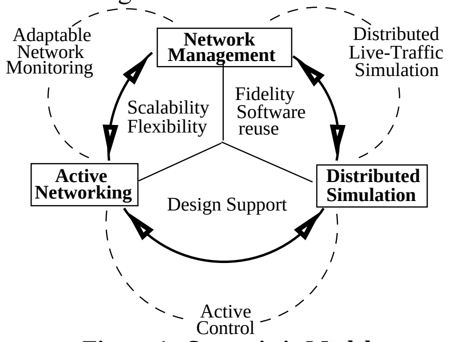

Figure 1 Synergistic Model

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 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.

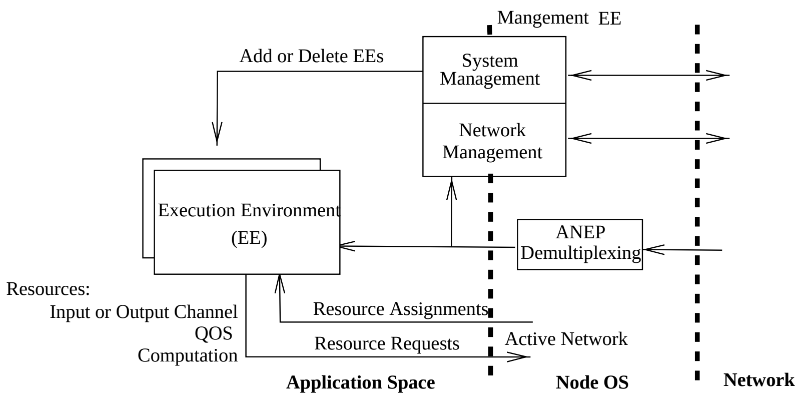

Figure 2 Active Network Node Architecture

This has led researchers to draft the active network node architecture depicted in Figure 2 (refer to [1] for more details). 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) [1] to the different EEs (including the developing new management and engineering technology that fits well within this design).

2.2 ANCORS Architecture

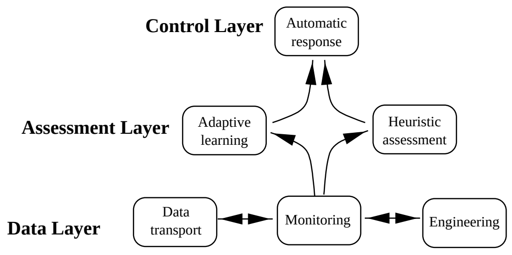

Figure 3 ANCORS Architecture

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 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. Figure 3 shows how the data and information flow through the layers. 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.

The ANCORS architecture has been designed to reuse and integrate software components derived from significant advances in network alarm correlation, fault identification, and distributed intrusion detection. In particular, the assessment and control layers of the ANCORS architecture perform tasks analogous to alarm correlation and fault analysis of the types currently proposed by network management expert systems [3,14,18].

All the components constituting these logical layers may be independently deployed and configured on machines throughout the network using common system management support. ANCORS may distribute data-layer services on machines across domains,[2] but deploys assessment and control layer services in machines within the 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 machines in 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 the data layer, which embodies the most innovative features of our 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 into three distinct data service types, all of which may benefit from dynamic deployment in the network.

2.2.1.1 Data Transport Services

Data transport services offer communication protocols that are either quite general and extensible as proposed by active networking concepts, 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.

[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.

2.2.1.2 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 report relevant information to higher-layer components in the ANCORS architecture. A variety of legacy SNMP- or CMIP-based agents, such as 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.

The use of active networking to dispatch user-definable monitoring capabilities gives ANCORS two major advantages: (1) it permits selective monitoring of particular phenomena, such as new network requirements and new usage patterns that emerge over time, and (2) it improves monitoring scalability (as suggested in [8,21]) through various degrees of sophistication in the monitoring agents, thus allowing, as suggested in [6], 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. These services are deployable in the NM EE of the active network node discussed in Section 3.

2.2.1.3 Engineering Services

Engineering services aid in the design and testing of network services before their deployment. 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 time scale that may differ from physical time,[4] and (3) they generate synthetic network traffic that does not contain user data.

ANCORS incorporates a distributed simulation engineering service to help operators explore and select the optimal deployment and configuration of network assets. Like monitoring services, the ANCORS simulation component can be distributed to key traffic arteries of the network, and can perform high-fidelity protocol and network component simulations based on the content of selective live traffic. The results of these simulations can be used to better understand the affects of various alterations of network behavior (which may become commonplace under the active network paradigm), before committing the network to these alterations. Section 4 presents an overview of the ANCORS distributed simulation services.

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

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

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. 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. *Anetd* handles process control requests coming from the management stations or automatic response services to either load new services or terminate existing ones.

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 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]

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.

3.1.1 System Management Commands

Anetd system management 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] Under the current version of *Anetd*, these core functions are supported by nine system management commands: *load, configure, query, kill, put, get, init, getacl*, and *getweb*. While future extensions to this command set may be required as additional functionality is

[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.

[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.*

desired, our current experience in building and distributing *Anetd* clients indicates that these primitives provide sufficient coverage necessary for basic management. Because of space limitations, we will only describe in detail the LOAD command, which is the most important. (Refer to [11] for the other commands.)

The LOAD command instructs *Anetd* to download files specified via a 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 path 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, the Java's Virtual Machine automatically downloads the required bytecode from the specified HTTP server, following the Java's RMI specification [5].

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.

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; refer to [11] for a detailed description.

3.1.2 Deploying ANTS

We include an example demonstrating how a client can deploy *ANTS* [19] 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/ConfigManager
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/ConfigManager
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/ConfigManager
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 which, in this case, will independently receive packets on port 8000.

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 the ANCORS management services.

4 Distributed Simulation

Adaptable and configurable networks will require code repositories with which deployable applications can be stored and retrieved. This idea has already appeared in several network management designs where deployable monitors can be dynamically inserted to key points in a network. Under ANCORS we are reusing and extending these concepts in the development of generic and reusable simulation models, which are deliverable as part of an ANCORS simulation service. In particular, we are developing simulation models that allow network engineers to compose and design experiments dynamically, based on the content of network traffic observed from spatially distributed points in a network. The following briefly summarizes the benefits of extending simulation into the network management framework, and how issues of resource utilization can be controlled and balanced against the fidelity of simulation results.

4.1 Distributed Simulation and Network Management

As with any distributed application, network simulation experiments will require some form of remote management. In our paradigm, the network management infrastructure and protocols will be reused for this task. In particular, as explained in Section 3, the deployment and control of simulations is performed through the same mechanisms used to deploy and control both monitoring agents and communication protocols.

This work differs significantly from current simulation work in that the current model of simulation typically involves the generation of synthetic workloads derived from statistical models. Inevitably, these high-level models of network traffic fail to capture important phenomena of the real load experienced by the real network. By including simulation in the network management infrastructure, it is possible to feed real workloads to the simulation system and thus greatly improve its fidelity. In some cases the workload abstracted from monitoring agents may be directly piped to the simulation, while in other cases this may not be possible because of differences in time-scale. In either case, network management tools can be used to first define the workload parameters required by the simulation and then feed collected data to the design experiments.

4.2 Controlling Resource Utilization

Simulation is typically used to predict the performance of a design by abstracting the behavior and performance of the design. The abstraction is usually performed in a way that optimizes the use of computing and communication resources to address very specific design problems. Distributed simulation offers a way to divide the execution of the software across a network to exploit the model's parallelism. The amount of abstraction used in the simulated models has a huge impact on the amount of resources required to simulate a given system. For example, if one wanted to study the performance of a particular transport protocol, at one extreme one could simulate the protocol by producing the actual packets of the protocol and transmitting actual simulated data between the simulated hosts, or at the other extreme only transmit high-level digests of the packets to abstract connections or the number of bytes to transfer in each session.

4.2.1 Balancing Fidelity with Resource Availability

We are currently focused on high-fidelity protocol development and prototyping. This kind of engineering service may require a substantial amount of communication and computing resources to be effective. This use of engineering services (which is our focus in [9,12]) should either be relegated to dedicated portions of the network so as not to interfere with normal operations, or should be executed slower than as fast-as possible to limit the amount of resources used. It is important to realize that even when the simulation is relegated to specific designated areas of the network (for example, a LAN of low-cost PCs), the engineering support can still benefit from being part of an integrated system like ANCORS.

In the near future will also develop higher-level models that will be used to assess the efficacy of different routing protocols. These models will consist of simulated routing tables and protocols that update the routes dynamically to minimize delay. In this case the simulation would require few resources from the network (in the order of a few tens of packets every several seconds) and may very well be placed on actual network nodes. In this case the simulator, through data provided by an ad-hoc monitoring agent, would assess the relative benefit of higher levels of dynamism in the route updates.

4.2.2 Not as Fast as Possible

Most simulators run as fast as possible to accommodate the designers' needs. In some situations, however, as fast as possible may not be the best solution or may not be desirable. In some cases, simulation, although it could execute faster than the target system (i.e. the simulated time advances faster than the physical time) must be slowed down to the speed of physical time to allow humans to interact with it (for example, flight simulators). In some other cases, as in network engineering, the simulation may be slowed to prevent consuming too much of the computing and communication resources. The degree of slowdown is intimately tied to the amount of the resources one wants to dedicate to simulation relative to other network functions and should therefore be set accordingly. When we conduct our network engi-

neering experiments, we initialize this speed by calling a function devoted to regulate the speed of the simulation. The function has the form
hardware_synch(<virtual_time_ticks>);

Through this function the user can specify the maximum amount of virtual time ticks per second. Whenever any of the models engaged in a simulation start consuming too many resources and let simulation time advance too fast, they are suspended until they are allowed to execute more events. The range of values for *<virtual_time_ticks>* depends on the time granularity of the simulated system and therefore cannot be generalized. In our high-fidelity experiments, each virtual time tick corresponds to 10 μs and we therefore *set <virtual_time_ticks>* to 100,000 to pace the virtual time with physical time.

4.2.3 Software Emulation versus Simulation

As part of our research we are exploring how to support the analytical requirements necessary for design versus prototyping, and have developed some techniques [9,10] to offer a prototyping and engineering environment. Using our methodology, a designer can (1) build a distributed software emulation to first verify the correctness of the implementation or the ideas and (2) then later, if necessary, augment the model(s) with mechanisms to yield a distributed simulation capable of performing timing-based quantitative measurements.

Emulation reproduces the behavior of a design by substituting some of the components of the system that may be not available or enable a better development environment. Emulation only tests the behavioral semantic of a system but does not provide (in most cases) metrics that are related to time. For example, emulating a transport protocol may help in the development and prototyping of the finite state machinery but may not allow the measurement of how long each of the operations may take or the quantitative effects of contention on the transmission lines.

Simulation, on the other hand, has the notion of virtual time and can be used to estimate the time at which different operations take place in the design. Simulation can therefore be used to perform detailed quantitative analysis of a design's performance. Distributed simulation is typically much more expensive than distributed emulation because it must keep track of a global notion of virtual time among the simulating entities. This resource requirement gap between emulation and simulation can therefore be exploited to tailor the amount of resources dedicated to network engineering

5 Virtual Networking Using ANCORS

As an example of an engineering service we have developed several deployable components to both instantiate and execute distributed simulations through *Anetd.* A base component exports a set of primitives that 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 [9].

To date, we have also developed a representative example of an engineering network service that emulates a UNIX kernel. This service 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 (such as *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 [13].

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 [9]. 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. 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 [9], we measured their performance and scalability.)

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 standards and virtually unlimited extensibility to introduce more powerful management technologies as they become available.

We are currently bridging 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] Active Networks Working Group. Architectural framework for active networks. Technical Report *http://www.cc.gatech.edu/projects/canes/arch/archdraft.ps*, 1998.

[2] D. S. Alexander, M. Shaw, S. M. Nettles, and J. M. Smith. Active bridging. *Proceedings of the ACM SIGCOMM'97 Conference*, Cannes, France, September 1997.

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

[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] 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.

[6] I. Katzela, A.T. Bouloutas, and S.B. Calo. Centralized vs. distributed fault localization. *Integrated Network Management IV*, 1995.

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

[8] K. Meyer, M. Erlinger, C. Sunshine, G. Goldszmidt, and Y. Yemini. Decentralized control and intelligence in network management. *Integrated Network Management IV*, 1995.

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

[10] L. Ricciulli, J. Meseguer, and P. Lincoln. Distributed simulation of parallel executions. *Proceedings of the 29th Annual Simulation Symposium*, 1996.

[11]L. Ricciulli. Anetd: Active NETwork Daemon. Technical Report, Computer Science Laboratory, http://www.csl.sri.com/ancors/Anetd, SRI International, 1998.

[12] L. Riccuilli, P. Lincoln, and P. Kakkar. TCP SYN flooding defense. *Proceedings of Communication Networks and Distributed System Modeling and Simulation Conference*, San Francisco, 1999.

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

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

[15] J. Smith, D. Farber, C. A. Gunter, S. Nettle, M. Segal, W. D. Sincoskie, D. Feldmeier, and S. Alexander. Switchware: Towards a 21st century network infrastructure. *http://www.cis.upenn.edu/~switchware/papers/sware.ps*, 1997.

[16] A. van Hoff, J. Giannandrea, M. Hapner, S. Carter, and M. Medin. The HTTP Distribution and Replication Protocol. *http://www.marimba.com/standards/drp.html*, August 1997.

[17] A. van Hoff, H. Partovi, and T. Thai. Specification for the Open Software Description (OSD) Format. *http://www.microsoft.com/standards/osd/*, August 1997.

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

[19] D. J. Wetherall, J. V. Guttag, and D. L. Tennenhouse. ANTS: A toolkit for building and dynamically deploying network protocols. *Proceedings of IEEE OPENARCH'98*, 1998.

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

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