A Survey of Active Network Research

David L. Tennenhouse, *Massachusetts Institute of Technology*
Jonathan M. Smith, *University of Pennsylvania*
W. David Sincoskie, *Bell Communications Research*
David J. Wetherall, *Massachusetts Institute of Technology*
Gary J. Minden, *University of Kansas*

Abstract

Active networks are a novel approach to network architecture in which the switches of the network perform customized computations on the messages flowing through them. This approach is motivated by both lead user applications, which perform user-driven computation at nodes within the network today, and the emergence of mobile code technologies that make dynamic network service innovation attainable. In this paper, we discuss two approaches to the realization of active networks and provide a snapshot of the current research issues and activities.

Introduction – What Are Active Networks?

In an *active network*, the routers or switches of the network perform customized computations on the messages flowing through them. For example, a user of an active network could send a "trace" program to each router and arrange for the program to be executed when their packets are processed. Figure 1 illustrates how the routers of an IP network could be augmented to perform such customized processing on the datagrams flowing through them. These active routers could also interoperate with legacy routers, which transparently forward datagrams in the traditional manner.

These networks are *active* in the sense that nodes can perform computations on, and modify, the packet contents. In addition, this processing can be customized on a per user or per application basis. In contrast, the role of computation within traditional packet networks, such as the Internet, is extremely limited. Although routers may modify a packet's header, they pass the user data opaquely without examination or modification. Furthermore, the header computation and associated router actions are specified independently of the user process or application that generates the packet.

The concept of active networking emerged from discussions within the broad DARPA research community in 1994 and 1995 on the future directions of networking systems. Several problems with today's networks were identified: the difficulty of integrating new technologies and standards into the shared network infrastructure, poor performance due to redundant operations at several protocol layers, and difficulty accommodating new services in the existing architectural model. Several strategies, collectively referred to as active networking, emerged to address these issues. The idea of messages carrying procedures and data is a natural step beyond traditional circuit and packet switching, and can be used to rapidly adapt the network to changing requirements. Coupled with a well

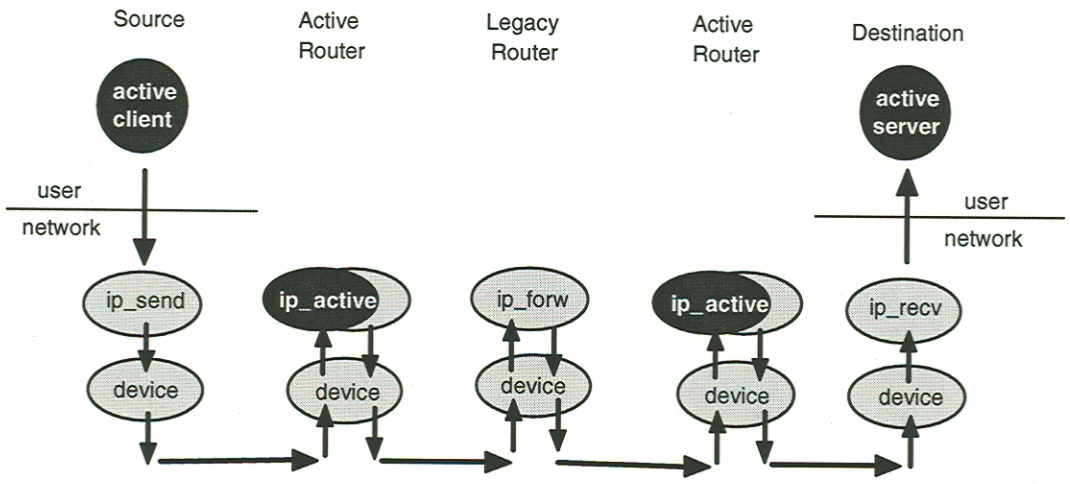

Figure 1. Application-specific processing within the nodes of an Active Network

understood execution environment within network nodes, this program-based approach provides a foundation for expressing networking systems as the composition of many smaller components with specific properties: services can be distributed and configured to meet the needs of applications; and statements can be made about overall network behavior in terms of the properties of individual components.

In this paper we discuss two approaches to the realization of active networks. The *programmable switch* approach maintains the existing packet/cell format, and provides a discrete mechanism that supports the downloading of programs. Separating the injection of programs from the processing of messages may be particularly attractive when the selection of programs is made by network administrators, rather than individual end users. In contrast, the *capsule* approach goes somewhat further – the passive packets of present day architectures are replaced by active miniature programs that are encapsulated in transmission frames and executed at each node along their path. User data can be embedded within these *capsules*, in much the way a page's contents are embedded within a fragment of PostScript code.

Research in active networks is motivated by both technology "push" and user "pull". The "pull" comes from the assortment of firewalls, web proxies, multicast routers, mobile proxies, video gateways, etc. that perform user-driven computation at nodes "within" the network. Some of these *lead users* are described in Table 1. In many cases, these services are implemented at nodes, such as firewalls, that adopt the facade of routers, yet perform application-specific processing that transcends conventional architectural guidelines. Our goal is to replace the numerous ad hoc approaches to network-based computation with a generic capability that allows users to program their networks.

The technology "push" is the emergence of *active* technologies that make our goals attainable. Until recently, the specter of administrators (let alone end users) programming their networks has raised insurmountable concerns with respect to infrastructure safety and efficiency. However, recent advances in programming languages, compilers and operating systems may provide the keys to the safe and efficient execution of mobile

CATEGORY	DESCRIPTION
Firewalls	Firewalls implement filters that determine which packets should be passed transparently and which should be blocked. Although they have a peer relationship to other routers, they implement application- and user-specific functions in addition to packet routing [1]. The need to update the firewall to enable the use of new protocols is an impediment to their adoption. In an active network, this process could be automated by allowing applications from approved vendors to authenticate themselves to the firewall and inject the appropriate modules into it.
Web Proxies	Web proxies provide a user-transparent service that is tailored to the serving and caching of Web pages. Harvest [2] employs a hierarchical scheme in which cache nodes are located near the edges of the network, i.e. within end user organizations. This system could be extended by allowing nodes of the hierarchy to be located at strategic points within the network.
Nomadic Routers	Kleinrock describes a "nomadic router" [3] that is interposed between an end system and the network. This module observes and adapts to the means by which the end system is connected to the network, e.g., through a phone line in a hotel room versus through the LAN in the home office. It might decide to perform more file caching or link compression when the end system is connected through a low bandwidth link and invoke additional security, such as encryption, when operating away from the home office.
Transport Gateways	"Transport gateways" are nodes located at strategic points that bridge networks with vastly different bandwidth and reliability characteristics, e.g., at the junctions between wired and wireless networks. To support mobile access to wired networks, TCP snooping [4] retains per-connection state information at wireless base stations.
Application Services	Application-specific gateways support services such as the transcoding of images [5] among video conference users with differing bandwidth constraints. Similarly, InfoPad [6] instantiates user-specific "pad servers", supporting applications such as voice and handwriting recognition, at intermediate nodes.

Table 1. Lead Users

program fragments. Today, these active technologies are applied within individual end systems and above the end-to-end network layer; for example, to allow web servers and clients to exchange Java applets. Active networks leverage and extend these technologies for use *within* the network – in ways that will fundamentally change our mindset concerning what is "in" the network.

This article provides a current snapshot of active network research activities, including work on the underlying active technologies. In the next two sections, we describe the impact active networks may have on infrastructure innovation and the new applications that will be enabled. We then present a framework, or set of issues, that can be used to categorize and organize activity within the field. Finally, we present a survey of current research activities within our own laboratories and elsewhere in the community.

Accelerating Infrastructure Innovation

As the *lead users* cited in Table 1 demonstrate, computation within the network is *already* happening – the demonstrated demand for these services suggests that network architectures must adapt to deal with this new reality.

At a more fundamental level, the network innovation process is itself ripe for renewal. The pace of network innovation is far too slow and, as the Internet grows, it is increasingly difficult to maintain, let alone accelerate, this pace. To a large degree this is a function of the need to achieve consensus – a network's utility increases with the number of interconnected nodes. Today, the path from prototype demonstration to large scale deployment takes about ten years. The process involves standardization, incorporation into vendor hardware platforms, user procurement and installation. The present backlog of Internet services includes multicast, authentication and mobility extensions, RSVP and IPv6.

The Internet Protocol (IP) enables interoperability by defining a standard packet format and addressing scheme; although router implementations may differ, they implement roughly "equivalent" programs. Thus, the mechanisms for IP innovation are: changing the IP service, which means changing everything (since it is the basis for interoperability); or establishing overlays, e.g. the MBone.

In contrast, active nodes can execute many different programs, i.e., they can perform very different computations on each of the packets flowing through them. Instead of insisting that all of the routers perform "equivalent" computations on every packet, active networks specify that all nodes support equivalent computational models, i.e., virtual instruction sets. Active networks raise the level of abstraction at which interoperability is realized, allowing applications to customize message processing to suit their purposes.

The ability to download new services into the infrastructure will lead to a user-driven innovation process in which the availability of new services will be dependent on their acceptance in the marketplace. Active networks present an opportunity to change the structure of the networking industry, from a "mainframe" mind-set, in which hardware and software are bundled together, to a "virtualized" approach in which hardware and software innovation are decoupled [7]. The network programming abstraction provides a powerful platform for user-driven customization of the infrastructure, allowing new services to be deployed at a faster pace than can be sustained by vendor driven consensus and standardization activities.

Enabling New Applications

Active Networks will enable new applications that rely on: the network-based merging of information; user-aware network protection; and active network management.

The merging and distribution of information

The era of multi-user, multi-site applications has just begun – the success of the MBone and the Web are but harbingers of what might lie ahead. There is an untapped reservoir of applications that require network-based services to support the merging and distribution of information. However, existing systems are based on a service that provides an extremely

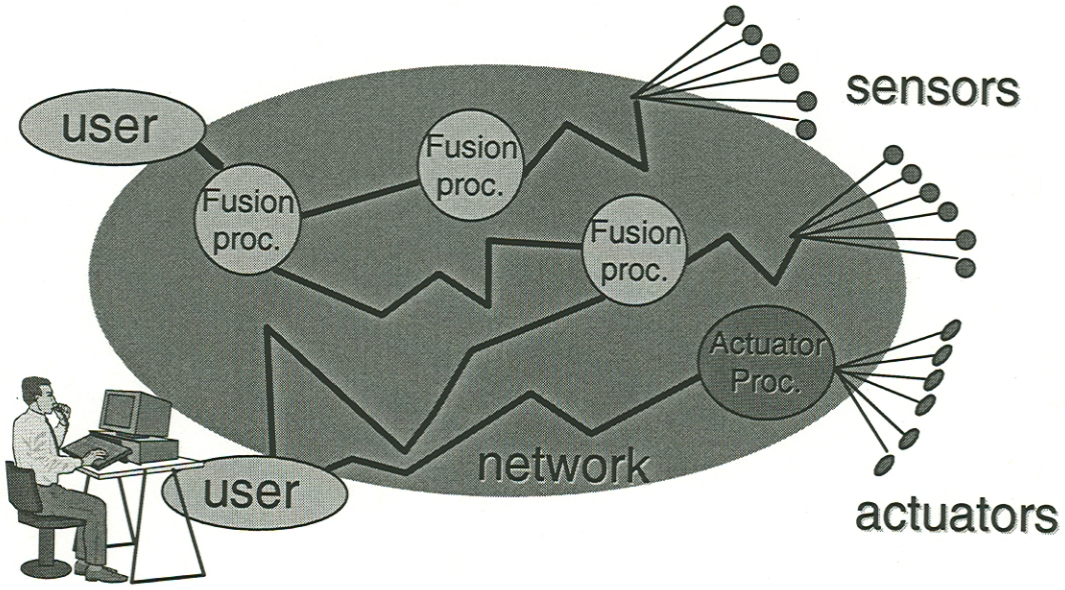

Figure 2. Exploiting the network-based merging and distribution of information. (Diagram courtesy of Prof. Henry Fuchs, UNC)

limited function, i.e., the copying of IP packets, without support for application-specific distribution, let alone network-based storage or information fusion.

Figure 2 illustrates how sophisticated multi-site applications will leverage computation and storage within the network. In this figure an application, such as simulation or remote manipulation, allows each user to "see" composite images constructed by fusing information obtained from a large number of sensors. Furthermore, each sensor can be "watched" by a number of users, who will have differing requirements concerning the encoding and presentation of the information they access. Merging data within the network reduces the bandwidth requirements at the users, who are located at the (low bandwidth) periphery of the network. Similarly, user-specific multicast services within the network reduce the load on the sensors and on the network backbone.

Web proxies that cache pages of information, are another example of a multi-user service that could benefit from network-based computation and storage. Harvest [2] employs a hierarchical caching scheme that can reduce the latencies experienced by individual users and the aggregate bandwidth that is consumed. The cache nodes are presently located near the edges of the network, i.e., at nodes within the end user organizations. These systems could be extended by allowing nodes of the hierarchy to be located at strategic points within the networks of Internet access providers and inter-exchange carriers. An interesting problem is the development of algorithms and tools that automatically "balance" the hierarchy by re-positioning the caches themselves, not just the cached information. A further argument in favor of using active technologies for Web caching is that a significant fraction of Web pages are dynamically computed and not susceptible to passive caching. This suggests the development of schemes that support active caches that store and execute programs that generate these pages.

User-aware network protection

Protection of information means that the right information gets to the right people at the right place and time. Although network security and authentication mechanisms are being proposed in many networking forums, active networking may admit the design of an integrated mechanism that governs all network resources and the information flowing through them. This eliminates the need for multiple security/authentication systems that operate independently at each communication protocol layer. It allows us to program in security policy for the network on a per-user or per-use basis. Finally, a formal approach using rigorous specifications and language enforced type-safety can be used to reason about the protection policies and the mechanisms of their implementation.

Active network management

Many network management tasks consist of collecting and collating data, such as event counts. To provide the most useful network management data, such as exception indications, intelligence must be used to filter out uninteresting (unexceptional) events. Active technologies could be used to implement sophisticated approaches to network monitoring and event filtering. Network components, such as routers, may even assume a degree of responsibility for monitoring themselves, e.g., by injecting customized monitoring and diagnostic programs into their nearest neighbors. Similarly, active networks can provide the flexibility necessary to improve fault detection and to update the *survivability* policies that govern component response to correlated failures, such as those caused by earthquakes or malicious intruders.

A Framework for Active Network Research

In this section, we distinguish two approaches to active networks, discrete and integrated, depending on whether programs and data are carried discretely, i.e., within separate messages, or in an integrated fashion. We then discuss common issues related to node programming and interoperability.

Programmable switches – a discrete approach

The processing of messages may be architecturally separated from the business of injecting programs into the node, with a separate mechanism for each function. This preserves the current distinction between in-band data transfer and out-of-band management channels. Users would first inject their custom processing routines into the required routers. Then they would send their packets through such "programmable" nodes much the way they do today. When a packet arrives at a node, its header is examined and the appropriate program is dispatched to operate on its contents.

Separate mechanisms for loading and execution might be valuable when program loading must be carefully controlled. Allowing operators to dynamically load code into their routers would be useful for router extensibility purposes, even if the programs do not perform application- or user-specific computations. In the Internet, for example, program loading could be restricted to a router's operator who is furnished with a "back door" through which they can dynamically load code. This back door would at minimum authenticate the operator and might also perform extensive checks on the code that is being loaded.

PROJECT	M	S	E	DESCRIPTION
Safe-Tcl [8] **(source)**	X	X		Safe-Tcl (based on Tcl) is a scripting language that provides safety through interpretation of a source program and closure of its namespace. It depends on the restricted closure and correctness of the interpreter to prevent programs from deliberately or accidentally straying beyond their permitted execution environment.
Java [9] **(intermediate)**	X	X	x	Java uses an intermediate instruction set to achieve mobility. Traditionally, the safe execution of intermediate code has relied on its careful interpretation. One of Java's key contributions is to improve efficiency by off-loading responsibility from the interpreter: the instruction set and its approved usage are designed to reduce operand validation per executed instruction. Work at the University of Arizona and elsewhere seeks to further boost efficiency through the use of compilation techniques.
Omniware [10] **(object-code)**	x	X	X	Omniware portable object-code depends on software-based fault isolation (SFI) to enforce safety efficiently. It prescribes a set of rules that instruction sequences must adhere to, e.g. restrictions on how address arithmetic is performed. In conjunction with run-time support, these rules define a "sandbox" within which the program can do what it likes, but that it may not escape.
Proof-Carrying Code [11] **(object-code)**		X	X	PCC uses a novel approach to achieve safety: it attaches a formal proof of the properties of a binary program. The recipient can check that the proof is valid, a process that is much simpler than constructing it from scratch. Currently, PCC is practical only for short programs.

Table 2. Program Encoding Technologies (with labeled columns M, S, and E assessing mobility, safety, and efficiency, respectively)

Capsules – an integrated approach

A more extreme view of active networks is one in which *every* message is a program. Every message, or capsule, that passes between nodes contains a program fragment (of at least one instruction) that may include embedded data. When a capsule arrives at an active node, its contents are evaluated, in much the same way that a PostScript printer interprets the contents of each file that is sent to it.

Bits arriving on incoming links are processed by a mechanism that identifies capsule boundaries, possibly using the framing mechanisms provided by traditional link layer protocols. The capsule's contents are then dispatched to a transient execution environment where they can safely be evaluated. We hypothesize that programs are composed of instructions, that perform basic computations on the capsule contents, and can also invoke "built-in" primitives, which may provide access to resources external to the transient environment. The execution of a capsule results in the scheduling of zero or more capsules for transmission on the outgoing links and may change the non-transient state of the node.

Towards a common programming model

Network programs must be transmitted across the communication substrate and loaded into a range of platforms. This suggests the development of common models for: the encoding of network programs; the "built-in" primitives available at each node; and the description and allocation of node resources.

Program encoding. Our objectives for program encodings are that they support:

- Mobility – the ability to transfer programs and execute them on a range of platforms.
- Safety – the ability to restrict the resources that programs can access.
- Efficiency – enabling the above without compromising network performance, at least in the most common cases.

Mobility may be achieved at several different levels of program representation: express the program in a high-level scripting language, e.g. Tcl; adopt a platform independent intermediate representation, typically a byte-coded virtual instruction set, e.g. Java; or transfer programs in binary formats, e.g., Omniware. Table 2 describes recently developed *enabling* technologies that support the safe and efficient execution of each level of program encoding. We expect that all three approaches will prove useful: source encodings support rapid prototyping; intermediate representations provide a compact and relatively efficient way to express short programs; and commonly used modules might best be expressed at the object-code level.

A possible approach to node interoperability would be to agree on an intermediate instruction encoding as the backstop for code mobility. Node implementors and users would be welcome to leverage alternative encodings, so long as they provide mechanisms through which an intermediate encoding of a program can be obtained or generated. Implementors may also leverage techniques such as dynamic ("on-the-fly") compilation that optimize common processing routines, both by converting portable representations to native ones, and by specializing programs to individual contexts. Operating system support for more specific strategies, such as "path"-based scheduling, protocol code reorganization, and low-level extensibility should also prove useful. Table 3 describes some of these compilation and operating systems technologies.

Common primitives. The services built-in to each node might include several categories of operations [12]: primitives that allow the packet itself to be manipulated, e.g., by changing its header, payload, length, etc.; primitives that provide access to the node's environment, e.g., the node address, time-of-day, link status, etc.; and primitives for controlling packet flow, such as forwarding, copying, discarding. Additional primitives might provide access to node storage and scheduling, e.g., to facilitate rendezvous operations that combine processing across multiple packets.

Node resources and their allocation. Beyond encodings and primitives, there must be a common model of node resources and the means by which policies governing their allocations are communicated. The resources to be modeled include: physical resources, such as transmission bandwidth, processing capacity, and storage; as well as logical resources, such as routing tables and the node's management information base. Safe resource allocation is an area that will require considerable attention. Active nodes will be embedded within the shared network infrastructure, and so their designs must address a

PROJECT	DESCRIPTION
Scout [13]	Scout is designed to support communication-oriented tasks. It allocates and schedules resources on a "path" basis and applies a number of optimizations intended to increase throughput and decrease latency. Many of the techniques may be applicable to programs loaded into network nodes.
Exokernel [14]	The exokernel enables programs to safely share low-level access to system resources. It implements a thin veneer that securely multiplexes the raw hardware. This in turn allows programs to tailor their own abstractions of operating system services, e.g., access to the active node environment.
SPIN [15]	SPIN relies on the properties of the Modula 3 language and a trustworthy compiler to generate programs that will not stray beyond a restricted environment. Programs signed by the compiler may be dynamically loaded into the operating system.
'C [16]	'C and VCODE enable "on-the-fly" code generation. This allows source programs to be automatically tailored, or even wholly generated, at runtime. These technologies could allow active nodes to translate commonly-used programs to binary encodings.

Table 3. Operating System Technologies

range of "sharing" issues that are often brushed over in the design of programmable systems destined for less public environments.

Current Research

Work on active networks is underway at a number of sites that are independently studying: capsule and programmable switch architectures; enabling technologies; specification techniques; end system issues; and applications, including network management, mobility and congestion management.

Massachusetts Institute of Technology

The MIT team is prototyping an architecture based on the capsule approach [17] and studying issues related to component specification, "active storage", multicast NACK fusion, and network-based traffic filtering. A reference platform that demonstrates the capsule architecture is being implemented on Linux using a Java-based capsule encoding. Additional enabling technologies, including advanced operating system techniques [14] and "on-the-fly" compilation [16] are also under investigation.

Capsules use the built-in constructs of a programming language to perform packet processing. This language will be extended through the specification of a suite of "foundation components" that invoke built-in primitives, interact with the local node environment, and can be extended and specialized to suit application-specific requirements. Demand loading and the caching of components are being developed as strategies to support compact programs and reduce the overhead associated with their transfer and evaluation. Demand loading allows capsules to reference components rather than carry them; and caching implies that recently used components need not be reloaded and verified for safety.

Programming will also be facilitated by allowing capsules to leave "soft state" behind in a node. Thus, a flow or connection may be opened by having a capsule leave a small amount of associated state at each node along the path it traverses. Subsequent packets can include code whose execution leverages this "soft state" but can regenerate it if necessary. Connections and flows in active networks can be more powerful than those of present day systems because the state left behind may be in the form of programs. A more persistent form of active storage, workflow state, is being developed to support loosely synchronized activities and to track dependencies.

University of Pennsylvania

The SwitchWare project [18] is developing a programmable switch approach that allows digitally signed type-checked modules to be loaded into the nodes of a network. The basic idea is to raise the level of abstraction of the switch functionality to be closer to that of a Turing machine. Aspects of security dictate limitations in the tradeoffs which can be made in support of other goals: resource allocation must be robust enough that denial of service attacks are frustrated; extensibility must be restricted so as to preclude security breaches, yet still adequate for advanced applications.

Penn's approach uses formal methodologies to prove security properties of SwitchWare programs. The focus of SwitchWare is the identification of properties of the underlying infrastructure for which theorems can be developed. Proofs are supported by a language (SML/NJ) with a precise definition and run-time support that includes concurrent garbage collection and resource allocation. An advantage of supporting security at the programming language level is that the high overhead of protection domain-crossing in kernelized operating systems is avoided, since the need for carefully gated entry points is removed at compilation time.

The approach will be evaluated with a prototype based on a shared-memory multiprocessor. Early prototype applications include: software scalable bandwidth based on a general mechanism for inverse multiplexing, i.e., network striping; and support for an active packet model ("Switchlets").

Bell Communications Research

Several aspects of the Penn design will be studied jointly with Bellcore, using a different infrastructure (OPCV2) to extend the design space that is explored. The Output Port Controller Version 2 (OPCV2) attaches to the Sunshine Asynchronous Transfer Mode (ATM) switch, developed for the AURORA Gigabit Testbed, and can also be used as a standalone cell processor that enables line speed manipulation of ATM streams. This allows studies of SwitchWare multiplexing algorithms and run-time system functionality to be embedded in the port controllers of a scalable switch.

A second component of the Bellcore effort is the specification of the semantics of an Active Router, and the investigation of those semantics in a collaboration prototyping effort involving Penn. The prototype will use a small-scale multiprocessor as an active network element that interconnects ATM networks with 10 and 100 Mbps Ethernets. This Active Router will serve as an experimental platform for the investigation of applications under development within the SwitchWare project.

Bellcore is also studying uses of the new network infrastructure, such as Self-Paying Information Transport, in which electronic payment information is embedded in the active packets. Bellcore's interest in active networks is related to its previous work on: protocol boosters [19], which dynamically optimize protocol components on an end-to-end basis; and the Advanced Intelligent Network (AIN), which separated the implementation of value added services from switching, by moving the service control functions to adjunct processors.

Columbia University

The NetScript project, led by Yemini and da Silva [20], consists of a programming language and execution environment. The language provides a means to script the processing of packet streams. It is particularly suited to the implementation of routing, packet analysis, signaling and management functions. NetScript agents can be sent to remote systems including intermediate network nodes, such as routers and switches. The goal is to enable programming of these nodes as easily and quickly as end-systems.

Carnegie Mellon University

The CMU team, led by Steenkiste and Zhang, is developing resource management mechanisms in support of "application-aware" networks. They are considering three dimensions of resource allocation: physical infrastructure, including processing and storage; decision making on different time scales, ranging from application startup to packet and cell scheduling; and the sharing of infrastructure among organizational entities. The mechanisms will support network customization across all three dimensions.

CMU is also exploring support for sophisticated multi-party applications, such as video conferencing and data mining, that use a multiplicity of traffic streams with divergent characteristics. These applications will be "network-aware" so they can perform well on a variety of networks and adapt quickly to changing network conditions.

Work Elsewhere

Additional research on active networks is being conducted at several sites:

- At BBN, Partridge and Jackson are exploring issues of programmability, data dictionaries, and authentication mechanisms, in the context of IP and to improve management and diagnostic capabilities.
- At the Georgia Institute of Technology, active network concepts are being applied to network congestion by allowing applications to request that specific node algorithms (e.g. lossless compression, selective discard, transcoding) be invoked during periods of congestion [21].
- At the University of Kansas, Frost and Minden are considering the application of active technologies to rapidly deployable radio networks.
- At the University of Arizona, Peterson is developing "liquid" software, a suite of mobile code technologies that includes rapid compilation of intermediate code, i.e., at network link rates [22].
- At the University of Cincinatti, Alexander is investigating techniques for the formal specification of network elements and behavior.

Summary

We realize that suggestions for software-intensive approaches to networking surface every ten years or so. For example, Zander [23] describes an experimental system in which packets of FORTH code were interpreted by network elements. Nonetheless, we are convinced that recent improvements in the safety and efficiency of active technologies, and the demand created by lead applications, present new research opportunities.

Active networks involve the synthesis and extension of programming language, operating systems and networking expertise. We also anticipate changes to the organization of end-system software – in place of protocol "stacks", applications may use protocol "components" that can be specialized and composed to perform application-specific functions [24]. This will lead to a massive increase in the degree and sophistication of network-based computation and address the mismatch between the rate at which user requirements change and the pace at which network infrastructure can be deployed.

Acknowledgments

This work has been influenced by discussions with a number of researchers, especially Deborah Estrin, David Farber, David Feldmeier, Henry Fuchs, Steve Garland, Carl Gunter, John Guttag, Frans Kaashoek, Butler Lampson, Paul Leach, Scott Nettles, Hilary Orman, Herb Schorr, Scott Shenker, Liuba Shrira and Bob Sproull.

References

1. Greenwald, M., *et al. Designing an Academic Firewall: Policy, Practice and Experience with SURF.* in *Proc. of the 1996 Symp. on Network and Distributed Systems Security*. 1996. San Diego, CA.

2. Chankhuntod, A., P.B. Danzig, and C. Neerdaels. *A Hierarchical Internet Object Cache*. in *Proceedings of 1996 USENIX*. 1996.

3. Kleinrock, L. *Nomadic Computing (Keynote Address)*. in *Intl. Conf. on Mobile Computing and Networking*. 1995. Berkeley, CA: ACM.

4. Balakrishnan, H., *et al. Improving TCP/IP Performance over Wireless Networks*. in *Intl. Conf. on Mobile Computing and Networking*. 1995. Berkeley, CA.

5. Amir, E., S. McCanne, and H. Zhang. *An Application Level Video Gateway*. in *ACM Multimedia '95*. 1995. San Francisco, CA.

6. Le, M.T., F. Burghardt, and J. Rabaey. *Software Architecture of the Infopad System*. in *Mobidata Workshop on Mobile and Wireless Information Systems*. 1994. New Brunswick, NJ.

7. Tennenhouse, D., *et al.*, *Virtual Infrastructure: Putting Information Infrastructure on the Technology Curve.* Computer Networks and ISDN Systems, Vol. 26, No. 13. (October 1996)

8. Borenstein, N. *Email with a Mind of its Own: The Safe-Tcl Language for Enabled Mail*. in *IFIP International Conference*. 1994. Barcelona, Spain.

9. Gosling, J. and H. McGilton, *The Java Language Environment: A White Paper*. 1995, Sun Microsystems: Mountain View, CA.

10. Adl-Tabatabai, A., *et al. Efficient and Language-Independent Mobile Programs*. in *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '96)*. 1996. Philadelphia, PA: ACM.

11. Necula, G. and P. Lee. *Safe Kernel Extensions Without Run-Time Checking*. in *Second Symp. on Operating System Design and Implementation (OSDI '96)*. 1996. Seattle, WA.

12. Wetherall, D. and D. Tennenhouse. *The ACTIVE IP Option*. in *7th ACM SIGOPS European Workshop*. 1996. Connemara, Ireland.

13. Mosberger, D. and L. Peterson, *Making Paths Explicit in the Scout Operating System*. 1996, Technical Report 96-05, Dept. of Computer Science, University of Arizona.

14. Engler, D.R., M.F. Kaashoek, and J. O'Toole, Jr. *Exokernel: An Operating System Architecture for Application-Level Resource Management*. in *15th ACM Symp. on Operating Systems Principles*. 1995.

15. Bershad, B., *et al. Extensibility, Safety and Performance in the SPIN Operating System*. in *15th ACM Symp. on Operating Systems Principles*. 1995.

16. Engler, D.R., W.C. Hsieh, and M.F. Kaashoek. *`C: A Language for High-Level, Efficient, and Machine-Independent Dynamic Code Generation*. in *23rd Annual ACM Symp. on Principles of Programming Languages (POPL '96)*. 1996. St. Petersburg, FL.

17. Tennenhouse, D. and D. Wetherall. *Towards an Active Network Architecture*. in *Multimedia Computing and Networking (MMCN 96)*. Jan 1996. San Jose, CA: SPIE. A revised version of this paper appears in Computer Communication Review, Vol. 26, No. 2 (April 96).

18. Smith, J., *et al.*, *SwitchWare: Accelerating Network Evolution*. 1996, Technical Report MS-CIS-96-38, CIS Department, University of Pennsylvania. Also available as http://www.cis.upenn.edu/~jms/white-paper.ps.

19. Feldmeier, D., A. McAuley, and J. Smith, *Protocol Boosters*. 1997. Submitted to IEEE Journal on Selected Areas of Communication. Jan 1997.

20. Yemini, Y., and da Silva, S. *Towards Programmable Networks*. 1996. in *FIP/IEEE International Workshop on Distributed Systems*. Oct 1996.

21. Bhattacharjee, B., K. Calvert, and E. Zegura, *An Architecture for Active Networking*, 1996. Technical Report GIT-CC-96-20, College of Computing, Georgia Institute of Technology.

22. Hartman, J., *et al., Liquid Software: A New Paradigm for Networked Systems.*. 1996, Technical Report 96-11, Dept. of Computer Science, University of Arizona.

23. Zander, J. and R. Forchheimer. *SOFTNET - An approach to high level packet communication*. in Proc. of AMRAD Conf. 1983. San Francisco, CA.

24. Clark, D.D. and D.L. Tennenhouse. *Architectural Considerations for a New Generation of Protocols*. in *SIGCOMM'90*. 1990.

Biographies

David L. Tennenhouse is a Principal Research Scientist at the MIT Laboratory for Computer Science and the Sloan School of Management. He is leader of the Telemedia, Networks and Systems Group, which is addressing "systems" issues arising at the confluence of broadband networks, advanced digital video, and distributed computing. David has a B.A.Sc. and an M.A.Sc in Electrical Engineering from the University of Toronto and a Ph.D. from the University of Cambridge. His is a member of the IEEE and his email address is dlt@lcs.mit.edu.

Jonathan M. Smith is an Associate Professor in the CIS Department of U. Penn., where he is a member of the Distributed Systems Laboratory. He was Co-Principal Investigator on the NSF/DARPA-sponsored AURORA gigabit testbed project. He is an editor of ACM/IEEE Transactions on Networking and a Senior Member of the IEEE. Jonathan previously worked for Boeing Aerospace, Bell Telephone Laboratories, and Bell Communications Research (Bellcore).

W. David Sincoskie is Vice President of the Internet Research Laboratory at Bellcore. He supervises research on broadband services control, gigabit networking, feature interaction, IP over ATM, NSFNET, next generation IP, and computer network management. He is a Senior Editor of JSAC and a Fellow of the IEEE. He was a member of the Internet Architecture Board from 1993 to 1995. David received his B.E.E. in 1975, M.E.E. in 1977, and Ph.D. (E.E.) in 1980 from the University of Delaware.

David J. Wetherall is a doctoral candidate at the MIT Laboratory for Computer Science. He is interested in networks, programming languages and distributed systems. David joined the Laboratory after working at QPSX Communications, a high speed networking company. He received his E.E and M.S. in computer science from MIT in 1995 and 1994, respectively, and his B.E. in electrical engineering from the University of Western Australia in 1989. He is a student member of the IEEE.

Gary J. Minden is an Associate Professor in the EECS Department of the University of Kansas. He recently completed a tour at the Defense Advanced Research Projects Agency (DARPA) Information Technology Office. Gary's research interests are in the areas of large scale distributed systems which encompass high performance networks, computing systems, and distributed software systems. He received the B.S.E.E. degree in 1973 and the Ph.D. degree in 1982, both from the University of Kansas, and is a member of the IEEE.