Towards an Active Network Architecture

David L. Tennenhouse and David J. Wetherall*
Telemedia, Networks and Systems Group, MIT

ABSTRACT

Active networks allow their users to inject customized programs into the nodes of the network. An extreme case, in which we are most interested, replaces packets with "capsules" – program fragments that are executed at each network router/switch they traverse.

Active architectures permit a massive increase in the sophistication of the computation that is performed within the network. They will enable new applications, especially those based on application-specific multicast, information fusion, and other services that leverage network-based computation and storage. Furthermore, they will accelerate the pace of innovation by decoupling network services from the underlying hardware and allowing new services to be loaded into the infrastructure on demand.

In this paper, we describe our vision of an active network architecture, outline our approach to its design, and survey the technologies that can be brought to bear on its implementation. We propose that the research community mount a joint effort to develop and deploy a wide area ActiveNet.

1. INTRODUCTION

Traditional data networks passively transport bits from one end system to another. Ideally, the user data is transferred opaquely, i.e., the network is insensitive to the bits it carries and they are transferred between end systems without modification. The role of computation within such networks is extremely limited, e.g., header processing in packet-switched networks and signaling in connection-oriented networks.

Active Networks break with tradition by allowing the network to perform customized computations on the user data. For example, a user of an active network could send a customized compression program to a node within the network (e.g., a router) and request that the node execute that program when processing their packets. These networks are "active" in two ways:

- Switches perform computations on the user data flowing through them.
- Individuals can inject programs into the network, thereby tailoring the node processing to be user- and application-specific.

We have identified several architectural approaches to active networks. One approach, which we find particularly interesting, replaces the passive packets of present day architectures with active "capsules" – miniature programs that are executed at each router they traverse. This change in architectural perspective, from passive packets to active capsules, simultaneously addresses both of the "active" properties described above. User data can be embedded within these mini-programs, in much the way a page's contents are embedded within a fragment of PostScript code. Furthermore, capsules can invoke pre-defined program methods or plant new ones within network nodes.

Our work is motivated by both technology "push" and user "pull". The technology "push" is the emergence of "active" technologies, compiled and interpreted, supporting the encapsulation, transfer, interposition, and safe and efficient execution of program fragments. Today, 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 program fragments. Our innovation is to leverage and extend these technologies for use within the network – in ways that will fundamentally change today's model of what is "in" the network.

The "pull" comes from the ad hoc collection of firewalls, Web proxies, multicast routers, mobile proxies, video gateways, etc. that perform user-driven computation at nodes "within" the network. Despite architectural injunctions against them, these nodes are flourishing, suggesting user and management demand for their services. We are developing the architectural support and common programming platforms to support the diversity and dynamic deployment requirements of these "interposed" services. Our goal is to replace the numerous ad hoc approaches to their implementation with a generic capability that allows users to program their networks.

There are three principal advantages to basing the network architecture on the exchange of active programs, rather than passive packets:

- Exchanging code provides a basis for adaptive protocols, enabling richer interactions than the exchange of fixed data formats.
- Capsules provide a means of implementing fine grained application-specific functions at strategic points within the network.

---

*{dlt,djw}@lcs.mit.edu. http://www.tns.lcs.mit.edu/. An earlier version of this paper was delivered during the keynote session of Multimedia Computing and Networking, San Jose, CA, January 1996.

- The 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 standardization processes.

This paper presents our vision of an active network architecture and the approach we are following towards the deployment of an operational ActiveNet. The active network approach opens a Pandora's box of safety, security, and resource allocation issues. Although we do not present a complete design, we identify a number of specific research issues, outline the approach we are following towards their resolution and identify the technologies we intend to leverage. Our plan is to bootstrap a wide area ActiveNet using similar techniques to those used by the prototype MBONE, i.e., by locating platforms at strategic locations and "tunneling" through existing transmission facilities, such as the Internet.

In the next section we provide a description of some of the "lead user" applications that motivate an architecture that facilitates computation within the network. In section 3, we provide an overview of active networks, a high-level perspective on how we propose to organize their platforms and an introduction to the research issues that must be addressed. Section 4 describes the "instruction set" issues associated with an interoperable programming model and how "active technologies" can be leveraged to effect the safe and efficient evaluation of capsules. We then discuss the management of node resources, such as storage and link bandwidth, followed by our plan for the deployment of a research ActiveNet. We realize that our work challenges some key assumptions that have guided recent networking research and so the final sections of this paper discuss the architectural and structural questions raised by our approach.

2. LEAD USERS

Recently, there has been considerable interest in: agent technologies, which allow mobile code to travel from clients to servers; and in Web applets, which allow mobile code to travel from servers to clients. Active networks bridge this dichotomy by allowing applications to dispatch computation to where it is needed.

We are encouraged by the observation that a number of lead users have pressing requirements for the transparent interposition of computation within the network. These include the developers of:

- Firewalls, which are typically located at administrative boundaries.
- Web proxies and other services, such as DNS and multicast routers, that form strategic vertices of copy, fusion and cache "trees".
- Mobile/Nomadic gateways, placed near the edges of the network where there are significant discontinuities in the available bandwidth, e.g., the base stations of wireless networks.

These lead applications demonstrate that there is user "pull" towards active networks. In the absence of a coherent approach to interposition they have adopted a variety of ad hoc strategies. In many cases the interposed platforms present the facade of network layer routers, but actually perform application- or user-specific functions. Active networks will rationalize these diverse activities by providing a uniform platform for network-based computation.

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. The need to update the firewall to enable the use of new applications 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 are an example of an application-specific service that is tailored to the serving and caching of World Wide Web pages. Harvest [1] employs a hierarchical scheme in which cache nodes are located near the edges of the network, i.e., within the end user organizations. This system is scalable and could be extended by allowing nodes of the hierarchy to be located at strategic points within the networks of the 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. Schemes such as dynamic hierarchical caching [2] and geographical push-caching [3] begin to address this issue.

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 traditional (passive) caching. This suggests the development of web proxy schemes that support "active" caches that store and execute the programs that generate web pages.

Mobile/Nomadic Computing

Interposition strategies are used by a number of researchers addressing mobility. For example, Kleinrock [4] describes a "nomadic router" 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/or invoke additional security, such as encryption, when operating away from the home office.

Similarly, "nomadic agents and gateways" [4] are nodes that support mobility. They are located at strategic points that bridge networks with vastly different bandwidth and reliability characteristics, such as the junctions between wired and wireless networks. Application-neutral work on TCP snooping [5] improves the performance of TCP connections by retaining per-connection state information at wireless base stations. Application-specific services performed at gateways include file caching and the transcoding of images [6]. The InfoPad [7] takes the process even further, by instantiating user-specific "pad servers" supporting a range of applications, such as voice and hand-writing recognition, at intermediate nodes.

New Application Domains

There is an untapped reservoir of applications that require sophisticated network-based services to support the distribution and fusion of information. One promising direction is the development of multi-point communication strategies that are more flexible than the existing IP multicast service, which performs a very limited computation on the user data, i.e., copying. Application-specific multicast, for example, would provide the mechanism to realize the quality of service filtering suggested in [8] for video-conferencing.

Information fusion is an example of a domain that may leverage interposed computation. Applications such as sensor fusion, simulation and remote manipulation, allow users to "see" composite images constructed by fusing information obtained from a number of sensors. Fusing data within the network reduces the bandwidth requirements at the users, who are located at the periphery of the network. Placing application-specific computation near where it is needed also addresses latency limitations by shortening the critical feedback loops of interactive applications.

3. ACTIVE NETWORKS

In this section, we provide an overview of active networks – highly programmable networks that perform computations on the user data that is passing through them. 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 provide a high-level description of how active nodes might be organized and describe a node programming model that could provide the basis for cross-platform interoperability.

3.1 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. Users would send their packets through such a "programmable" node much the way they do today. When a packet arrives, its header is examined and a program is dispatched to operate on its contents. The program actively processes the packet, possibly changing its contents. A degree of customized computation is possible because the header of the message identifies which program should be run – so it is possible to arrange for different programs to be executed for different users or applications.

The separation of program execution and loading might be valuable when it is desirable for program loading to be carefully controlled or when the individual programs are relatively large. This approach is used, for example, in the Intelligent Network being standardized by CCITT. In the Internet, 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. Note that 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.

3.2 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.

Figure 1 provides a conceptual view of how an active node might be organized. 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 dispatched to a transient execution environment where they can safely be evaluated. We hypothesize that programs are composed of "primitive" instructions, that perform basic computations on the capsule contents, and can also invoke external "methods", 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. The transient environment is destroyed when capsule evaluation terminates.

3.3 Programming With Capsules

Our distinction between the discrete and integrated approaches is one of perspective, primarily useful as a basis for comparing two ways of thinking about networks and their programming. In practical terms, a network based on the integrated approach could be programmed to emulate the discrete approach and vice-versa. Nonetheless, we are intrigued by the