Department of Computer Science Technical Reports

Department of Computer Science

1997

Active Gateway: A Facility for Video Conferencing Traffic Control

Shunge Li

Bharat Bhargava
Purdue University, bb@cs.purdue.edu

Report Number:
97-011

ACTIVE GATEWAY:
A FACILITY FOR VIDEO
CONFERENCING TRAFFIC CONTROL

Shunge Li
Bharat Bhargava

Department of Computer Sciences
Purdue University
West Lafaytte, IN 47907

CSD-TR 97-011
February 1997

Active Gateway:
A Facility for Video Conferencing Traffic Control*

Shunge Li and Bharat Bhargava
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, U.S.A
{lis,bb}@cs.purdue.edu

Abstract

Active technology is emerging as one of the most promising fields in networking and distributed computing. It permits customized and more flexible control over the underlying networks. End users can systematically and strategically study and change the network behaviors by programming individual nodes of the network. The functionality of an active network goes beyond the traditional data transmission; each node of an active network has the ability to executing certain programs (or commands) passing through it. This paper describes our research work of applying active technology to video conferencing traffic control. We present the architecture and the design of an application-level facility, called active gateway, for video conferencing traffic and quality of service (QoS) control. We show that this facility enables more control functions for video conferencing which are not seen in conventional video conferencing tools. We illustrate its features through several experiments. Many important issues related to the design of an active network are discussed. Some technical solutions and future improvement are suggested.

1 Introduction

Recent technological advances in computing has come to a point where human's understanding of computation model and the way computing should be done are about to change. Computers are not stand-alone entities any more. They are networked globally, or at least within an enterprise or an organization. On the other hand, people have realized that the computer network itself

*This research is supported by NSF under grant number NCR-9405931

can function as a computer. It is no longer just a passive medium for moving data from one place to another. It can actively generate, store, exchange, and control information, and can be programmed by end users. In other words, it can *be* a computer itself!

Active technology, under this situation, is emerging as one of the most promising fields in distributed computing. Under the networking context, each network node is active if it can not only perform its traditional task of data transmission, but also execute programs or commands passing through it. Therefore, end users of an active network can systematically and strategically study and change the network's cooperative or non-cooperative behaviors by, for example, injecting to each active node a piece of programs in certain way and letting them be executed simultaneously to create some useful scenarios that aren't easily repeated in practice. With active technology, end users can have more flexible control of the network by programming the network as a whole. They can conduct scientific experiments repeatedly in a more controllable fashion. They can also customize the network computing environments by manipulating the network resources as well as the performance parameters of each node. By programming the network routers, end users can have better control over the way shared system and network resources are managed. Bandwidth mismatch problem among different network links, personalized QoS requirements, as well as traffic scheduling, can be handled more easily.

Active network or programmable network also brings to the network community another benefit that isn't seen in traditional network systems. Currently, it is difficult to incorporate new algorithms or services into routers or switches without going through the complex standardization process that can last for at least 5 years. This is due to the fact that whatever new services come to the intermediate gateways are expected to have a wide range of applications to build upon and therefore must be adopted with care. Active network permits rapid development, deployment, and utilization of new network services, protocols, functionalities by giving users the ability to programming network nodes and to experimenting their new ideas right away.

Active technology opens up a door with many opportunities in networking. For example, resource management is a big issue in multimedia communications. Resource reservation usually results in resource under-utilization. With active technology, bandwidth which has been reserved for some application can be temporarily *borrowed* by other applications needing it if the former is not using full amount of reserved bandwidth. This can be easily achieved by forcing the involved gateway to run a program that properly marshals the bandwidth among applications.

Active network is a timely service because its ability to executing programs at each node is much needed by many application domains: security, multimedia, network management, traffic control, service customization, to name a few. This paper describes our research work of applying active technology to video conferencing control. We present the architecture and the design of an application-level facility called active gateway. We show that its facility enables more control functions which are not seen in conventional video conferencing tools. We illustrate its features through several experiments. Many important issues related to the design of an active network are discussed. Some technical solutions and future improvement are suggested.

2 Related Work

An active network isn't something that is different from current networks in its entirety. Some forms of preliminary active network capabilities can be found in several products or systems. For example, a typical commercial router encapsulates multiple protocol modules that can be easily configured by system administrators based on their running environments. This kind of configurability is very limited compared to the programmability the active network offers, however, because human knowledge about when and what to configure in the router under a certain condition has not been incorporated into a program; the configuration task is currently performed through human interaction with the router at run time. With active network, we can program this human knowledge into the configuration task and launch the program at the intermediate routers. Routers will no longer need the system administrators to configure them; the program does it all.

Similarly, video gateway ([AMZ95]) and mixer mechanism defined in RTP ([SCFJ96]) can perform, at intermediate network nodes, certain application-level processing functions such as transcoding, stream synthesizing, and filtering. Firewalls, Web proxies, and agents are also mechanisms that allow fixed sets of processing tasks to be performed at some network nodes. The fact that all these technologies are applied within individual end systems and above the end-to-end network layer ([TW96]) has led to several recent research activities that leverage and extend these technologies for use *within* the network.

The laboratory for Computer Science at MIT has implemented a network service called *Active IP Option* ([WT96]), in which Tcl scripts are embedded in IP packets using IP option fields. Any gateway recognizing the special options can interpret and execute the scripts. Others just ignore them. With this new service, many traditional networking services, such as finding MTU of a remote network link, can be re-implemented in a more efficient fashion. More importantly, new functionalities can be built upon.

Researchers at MIT also proposed to build an experimental ActiveNet on the Internet ([TGSK96]). ActiveNet is intended to be designed as a network infrastructure that will serve as a testbed and environment for active network researches. It is proposed to be built very much the same way the MBone ([Eid94]) was built, that is, through tunneling. This is a cost-effective approach because it does not build a separate network. The ActiveNet will be made transparent to other Internet users as they won't be aware of the existence of the ActiveNet. The Internet and the ActiveNet will co-exist, and the Internet will operate as usual. About a dozen institutes have agreed to work towards this goal and many other organizations have expressed their interests in participating in such a construction.

NetScript ([YdS96]) is another effort by researchers at Columbia University towards building active programmable networks. The NetScript project focuses on the architecture issue for programmable network. Each intermediate programmable network node is considered as a Virtual Network Engine (VNE) interconnected by Virtual Links (VL). A VNE is programmed by NetScript agents to process packets and route them to other VNEs via some VLs. Agents themselves receive services from the Agent Services layer which supports delegation, execution, and control of agent programs, as well as communication services among agents. NetScript is a small and simple dataflow language specifically designed for communications-based tasks.

The idea of active network can be applied to network hardware facility as well. SwitchWare, proposed by researchers at University of Pennsylvania and Bellcore, is an instance of active network engineering practice [Sea96]. A SwitchWare is a programmable, software-controlled network switch intended to replace the special-purpose switching hardware. It consists of input and output ports controlled by a software-programmable element. Sequences of messages containing programs are sent to the SwitchWare switch's input ports, which interpret the messages as programs.

In this paper, we demonstrate a practical prototype system that controls video conferencing streams via active application gateways. We show that by programming active application gateways, we can achieve new control services that would otherwise hardly be achievable with traditional video conferencing technology.

3 Active Application Gateways

3.1 Design Methodology

3.1.1 Application Domains

One of the primary objectives for us to develop the facility of active gateways is to build a testbed for conducting experiments on real-time multimedia communications. This facility will allow us to study these issues in a more controllable and more systematic way. Study of new protocols for multimedia communications, new resource management schemes, and new QoS control mechanisms will be largely facilitated through the use of such a facility. However, the facility is not just limited to these domains. It can also be used in network management, distributed object systems, security, world-wide information systems, etc. In this paper, we focus on its use in video conferencing traffic control.

3.1.2 Kernel Implementation vs. Application-Level Implementation

We implemented the facility of active gateways as an application-level service because of the following reasons:

1. Because many features of active gateways are in experimental stage, they are expected to be added, removed, or updated frequently. Developing them directly within kernels would mean frequent kernel changes, compilations, and installations, which are time-consuming tasks.

2. An active network provides end users with an ability to customizing the systems for specific applications. A specific application, however, usually has specialized requirements that cannot be generalized to other applications. This highly application-oriented nature makes its development and deployment unsuitable to be done in the operating systems' kernels, which are supposed to offer services for *all* applications.

3. Developing active gateways as an application service allows current systems to continue operation without being affected by the development process.

4. Most developers don't have access to the kernel source codes, making application-level development the only choice.

Not only does the development of active gateways not need to change kernel, it does not require super-user permission to run it, either. In fact, during development process we deliberately avoided using those system services that would require root permission. A superuser permission requirement limits the usability of a software to a great extent. It may also leave some security holes to malicious users. Easy compilation, installation, execution, and great degree of usability are what we seek for this software.

3.1.3 Program Encoding Schemes

The scheme in which a program is encoded or expressed plays an important role in an active network. A good scheme must support mobility, portability, and efficiency. It may optionally support built-in security features, or simply let network protocols to handle the issue.

Mobility not only refers to heterogeneity or platform-independency, but also implies the ability to adjusting to running environments of different network nodes. Because a program is expected to flow through a number of network nodes on a range of possibly different platforms in an active network before it reaches its ultimate destination, either the program encoding scheme must be platform-independent or each active node must be capable of doing on-the-fly compilation on the program source codes. If the program is expressed in binary code, it may not be recognizable by some active nodes unless it also carries binary codes that can be recognized by the nodes.

However, simultaneously carrying binary codes for multiple platforms increases code redundancy and wastes considerable network bandwidth. Therefore, although binary encoding scheme is the most efficient one, it works only on platforms with binary compatibility. In a heterogeneous environment such as the Internet, this approach has very limited interoperability.

There are two ways to deal with programs expressed in source code scheme: interpretation and on-the-fly compilation. On-the-fly compilation, while achieving the greatest interoperability, has severe performance penalty because the nodes must first compile a program and link the object code with libraries before they can execute it. This time-consuming process is sometimes not affordable for real-time applications. Even though an application is not time-critical, run-time compilation usually takes long time, which will result in unpleasant response time. Interpretation, on the other hand, allows programs to be evaluated right away without requiring knowledge about the binary code formats of local platforms. For programs of small sizes, it has reasonable response time. Furthermore, it supports rapid prototyping.

Intermediate code scheme is a compromise between compilation scheme and interpretation scheme in terms of efficiency and portability. This approach introduces a concept of Virtual Machine (VM), whose intermediate code format is supported universally. Every active node recognizes the intermediate code format and has a translation routine that does the mapping from the intermediate code format onto binary code format of the local platform. A program is compiled and translated into the intermediate code format at the dispatcher node before dispatched to active nodes. Each active node then translates this intermediate code into its own binary code which is linkable and executable on its local machine. This approach achieves a good balance between portability and efficiency, therefore is the most promising approach. The only requirement for this approach to work globally is that every node must support VM intermediate code format (e.g., the bytecode format in Java) and must have a translation routine to map between its local platform binary code format and the VM intermediate code format.

Therefore, a good candidate of the program encoding schemes must be both platform-independent and interpretable. In addition, it must be simple and concise in order to save network bandwidth. Many scripting languages, such as Tcl ([Ous93]) and Perl ([Lar96]), meet these requirements. Java ([GM95]), though not a scripting language, falls into this category well due to its Virtual Machine (VM) model. Its nice integration with the World-Wide-Web (WWW) and its built-in security features make it especially attractive in future deployment of active network technology through WWW. Perl, as one of the major languages for Common Gateway Interface (CGI) programming in the WWW stage, has been very successful in the WWW world, which is by nature heterogeneous. It has great potential towards being developed as one major working language for active network. Tcl has already been used in the Active IP Option research. Its conciseness, ease to build GUIs with, and ease to integrate with C language, make it a non-negligible choice as a working language for active network. Other languages, such as ML, have also been used in active network programming ([Sea96]). [YdS96] developed a dataflow language called NetScript as a means for programmable network research. However, they need to be further developed towards mobility, portability, and efficiency, in order to be accepted globally.

We use Tcl scripting language in our development of active gateways due to following reasons:

1. Rapid prototyping. It took us about one month to build our prototyped (demonstratable) active gateways, though constant improvement is always needed.

2. Embeddability into C programming language, which is used to implement major gateway functionalities. Tcl's interpreter is implemented as a library of C functions that can easily be incorporated into applications.

3. GUI support by Tk, a companion toolkit of Tcl. Accessibility, customization, flexible control, and systematic manipulation of active network imply an easy-to-use and friendly GUI.

4. Efficiency and portability. For small programs, Tcl achieves reasonably good efficiency. Compared to Java-approach, which will be described later, no compilation process is involved. As to portability, Tcl's interpreters are available on most platforms.

3.2 Architecture of the Active Network

We have constructed an experimental active network prototype on top of our campus network environment. The active network consists of a set of active application gateways, a set of end systems, and a set of dispatchers, all of which can be placed on arbitrary network nodes. Such an active network is *virtual* because it does not exist physically; it is built on top of conventional IP networks. However, it *is* a network from end users' perceptions because they are not aware of the existence of the underlying physical network, which actually delivers dispatched programs transparently. They are aware only of the logical entities (e.g., routes, connections) in the active network. Creating such a virtual network is useful because it does not change the routers in the physical network at all and thus allows them to continue operation without being affected by the active network. Figure 1 depicts the relationship between an active (virtual) network and a physical network.

To avoid kernel changes within the routers of an active network and superuser permission requirement, we use UDP instead of raw IP as the basic networking service. Tcl scripts, along with other meta information (see the next subsection), are encapsulated in UDP datagrams, which are in turn delivered in conventional networks.

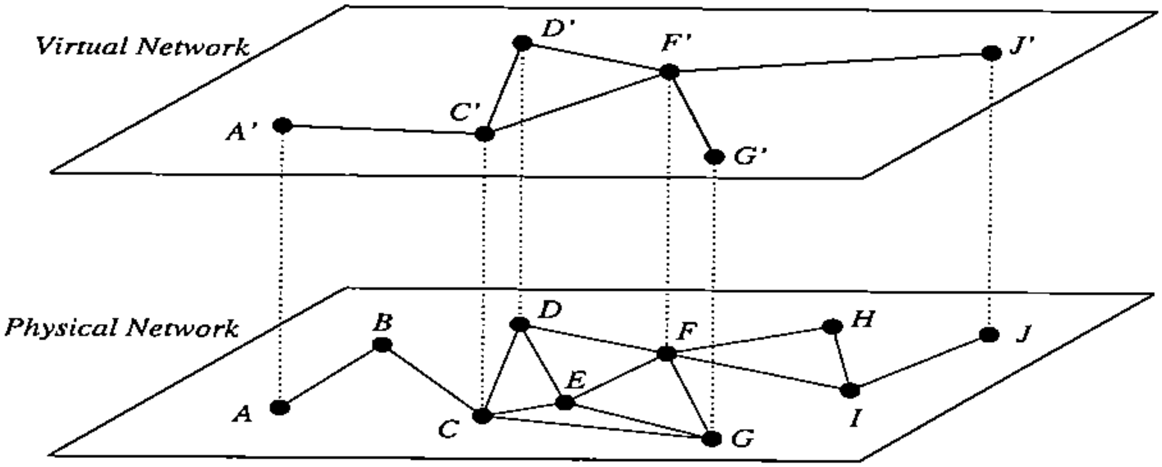

Figure 1: A Virtual Network and Its Physical Network

Figure 2 shows a simple but typical case of active network setup where an active gateway is placed in a network node between two networks each respectively connecting an end system. Also connected to the gateway via a separate network is a dispatcher, which serves as an operational platform for the active gateway. The dispatcher issues programs or scripts to the gateway over some network and receives from the gateway return messages that report the status of applications, resource usages, statistical information about network traffics, etc. The purpose of such a construction is to enforce the communication between two end systems to go through the gateway, therefore facilitating experimentation. The active gateway can perform traditional packet routing function in the logical sense in addition to executing programs dispatched from remote dispatcher softwares.

In general, an active network may contain arbitrary number of dispatcher programs and/or active gateways in the same physical network at the same time. They can provide services to arbitrary number of end systems and can communicate with each other using a built-in routing facility. Because the dispatcher programs and the active gateways are implemented at application-level, both can be launched as standalone applications from any nodes and run on any target machines in the network, allowing easy installation and utilization by end users. In practice, only one dispatcher is needed because it can concurrently dispatch various programs to as many gateways as possible.

3.3 Communication Facility

The communication facility in our active network is constructed to support dispatch of programs as well as communications among active gateways. An active gateway is responsible for normal packet routing as well as handling programs passing through it. A well known port number has

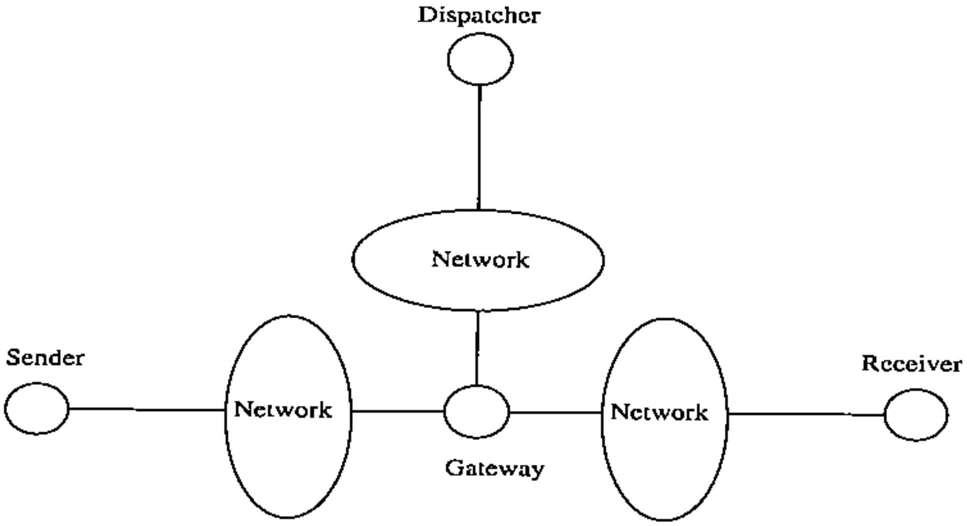

Figure 2: A Simple Active Network

been established for application-level packet routing. To demultiplex normal data packets and dispatched programs, we designate a separate port number to handle programs flowing through active gateways. Any active gateway listening to this particular port will receive program-embedded UDP datagrams passing through it.

Note that it is possible to share the same port between application data packets and program-embedded UDP datagrams. However, doing so would require to define a new *type* field in the packet header. For every application the active gateway wants to support, its packet structure has to be updated to incorporate this *type* field. Specifying a separate port number for dispatched programs would provide a more general mechanism that can work with as many applications as possible while minimizing modifications that have to be made to the applications.

Tcl scripts are encapsulated into UDP datagrams and dispatched to the network. Besides the scripts themselves, also included in UDP datagrams are extra control and meta information such as program encoding scheme, routing and addressing information, meta information about the scripts (length, constants, environment variables, parameters). For security reasons, the Tcl-scripts (or any other programs) need to be encrypted using some cryptographic scheme such as public key cipher. Meta information therefore may also contain digital signatures and public keys. These information are packaged as a message header of the Tcl scripts and are placed at a layer above UDP (table 1).

3.4 Components of Active Gateways

An active gateway is currently implemented with two threads. The Tcl interpreter runs as a thread and the router, which performs normal functions of routing, event handling, scheduling,

Table 1: Format of Script-Embedded UDP Datagrams Dispatched to Active Gateways

UDP Header	Meta Data	[Encrypted] Tcl Script

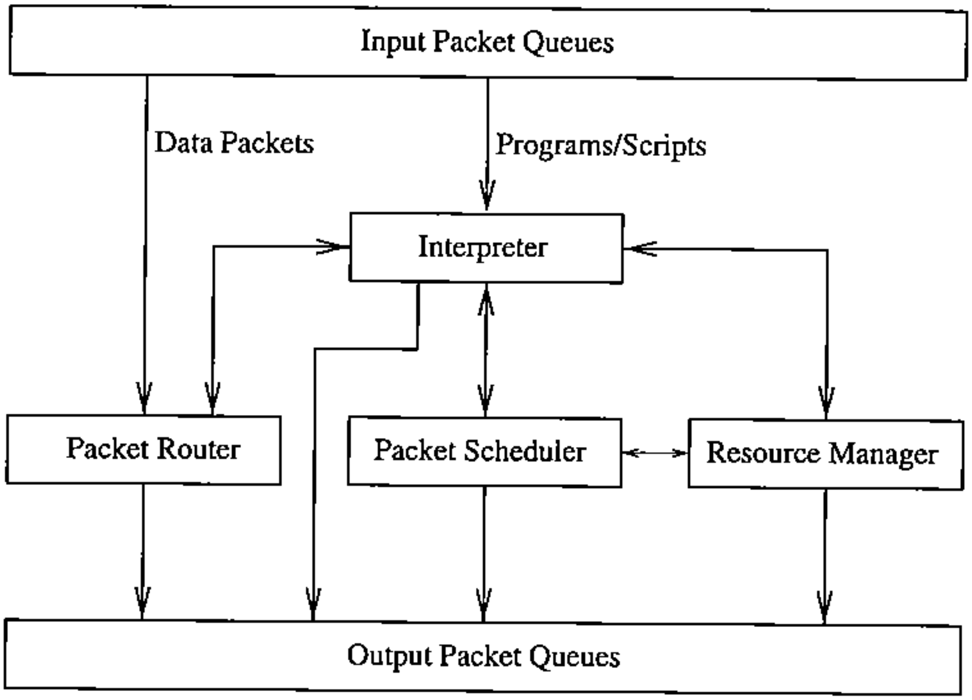

Figure 3: Components of An Active Gateway

runs as the other. Figure 3 shows the components of an active gateway.

3.4.1 Routing Tables

The most basic functionality a gateway performs is packet routing. Each active gateway has its own built-in routing table that is controllable by end users through program dispatching. This routing table specifies logical routes in the active network, not physical routes in underlying physical network. For example, one hop away (from *A'* to *C'*) in an active network may be several hops apart (from *A* to *C*) in an actual physical network (figure 1). Users of the active network are unaware of the routing operations performed in the underlying network; instead, they are aware only of the routing operations performed in the active network.

From implementation point of view, the routing table of an active gateway is no difference from that of a conventional IP gateway. For each destination IP address, it contains the IP address of the next hop gateway (in the logical sense). The routing table at each active node (hosts as well as intermediate gateways) is configurable through some primitives and is consulted to determine the next logical route to send the UDP datagrams to. The content of the routing table can be changed at any time by the active node by executing programs dispatched to it. This logical facility allows end users to easily perform application-specific traffic flow control without changing physical routes for other applications. Packets of other applications are routed the same way as if there were not active gateways in the network.

3.4.2 Primitives

Primitives are the smallest programmable units that can perform the most basic functionality for a specific application in an active network. They form a base from which all the application's control functions can be derived. In this paper, we are not intended to define primitives for the active network's control language. Rather, the primitives we are talking about here are application-oriented.

Our primitives can be categorized into two classes:

- **Communication primitives.** These are are basic operations involving communications among active network nodes. Examples include operations on routing table (setting, changing, or removing routes), network resources (setting bandwidth), network QoS (querying link latency, throughput), and application QoS (querying delay, jitter, connection information).

- **Non-communication primitives.** These are are basic operations that deal only with the local entities such as resources. Examples include operations on resource management, scheduling, priorization, policy, etc. Also provided are primitives regarding time event management, including invoking an event or a procedure at a particular time, delaying an event by a time interval, etc.

3.4.3 Policies

Policy-based control mechanism is a must for many distributed systems and applications. QoS control in real-time multimedia communications is by definition policy-based simply because there does not exist a single universal control scheme that works for all cases.

Policies used in active gateways can be classified based on different criteria. In terms of functionality, they can be classified as control policy and scheduling policy; in terms of the way they are defined and dispatched, they can be classified as static policy and dynamic policy; and in terms of generality, they can be classified as application-specific policy and general policy.

Control policies deal with how to perform a certain type of control. For example, a control policy may define *which* packets should be dropped or *which* connections (or channels) should be closed when the available network bandwidth falls below a threshold. Scheduling policies deal with how to schedule system and network resources. For example, a scheduling policy may state how to allocate and schedule resources among multiple connections. Static policies are predefined. Dynamic policies are defined and dispatched by end users on-the-fly.

Active gateways support many control policies and scheduling policies that can be either static or dynamic or both. During initialization, an active gateway loads predefined policies. During execution of a program, a newly defined policy can be taken into effect by replacing the current policy. This dynamic reconfigurability of policies can be programmed in Tcl scripts. Each policy has a unique policy number and is implemented as a Tcl procedure. Therefore, a policy can be overwritten by redefining the corresponding Tcl procedure on-the-fly.

3.4.4 GUI

A graphical user interface (GUI) has been designed and implemented for the dispatcher software to make its operations easy. It serves as a software platform for launching Tcl-scripts on active gateways. It greatly facilitates our experimentation.

Our GUI consists of a control panel (a set of control buttons/scales), an information panel reporting network status or statistical information, a text-editable window for users to type in commands/scripts and to dispatch to the gateways for execution. Menu bars and accelerate keys are also provided to facilitate and speed up operations. A graphical panel that can pictorially show state/status information is to be added to enhance the visualizability of information presentation.

3.4.5 Events

Many events are encountered in a real multimedia communication session. They include data loss, data corruption, and retard of data delivery. To study their effects on QoS, a mechanism is needed to control the generation of these events.

Active gateways support many operations other than routing. These *events* include dropping a packet, delaying the delivery of a packet by a certain amount of time, recording a packet, corrupting a packet, and transcoding a packet. The purpose of supporting these special operations at intermediate gateways is to simulate various network service models and traffic models on a more controllable network environment such as a campus network. Often, communications with UDP in LANs witness less or no data loss. Even in a lossy environment, the quantity of the loss rate is hard to control. By dropping packets flowing into an intermediate gateway with certain probability, we introduce a loss rate into packet delivery over a specific link. Also, many infrequently-occurred events can be re-produced as many times as we want. With this approach, coupled with active technology, data loss or other events becomes more controllable, manageable, and programmable.

An event generator has been implemented to generate these special events systematically. Associated with each event is a routine that implements the semantics of that event. These routines will be called when corresponding events are activated. The event generator follows a probabilistic model, that is, each event is generated with a certain probability. The probability distribution of these events is characterized by a probability distribution function, which can be either predefined or defined on-the-fly, or overwritten dynamically. A primitive (*set_pdf*) is created specifically for defining this function programmatically. The probability distribution functions can be per-application based or per-traffic flow based.

3.4.6 Priority

Priority is used to represent the relative importance among entities. It is the simplest and yet powerful mechanism for guaranteeing QoS such as end-to-end delay bound. Let's envision a scenario where a gateway is handling multiple incoming real-time and non real-time traffics concurrently. Obviously, assigning higher priority to real-time traffics over non real-time counterparts would give the former an ability to preempt shared resources and guarantee it to get scheduled/serviced within certain time bound to meet its deadlines.

Among the entities that can be prioritized are processes, connections (or channels), frames, and packets. The way in which priorities are assigned can vary. Policies can be defined to reflect different priority assignment schemes.

4 Experiments

4.1 Example Application – Video Conferencing

In previous section, we showed that active gateways support many mechanisms that are essential for providing control functions for distributed multimedia applications. For example, using the primitives provided, an active gateway allows resource utilization information such as buffer size, CPU time, and network bandwidth to be collected and the contents of routing tables to be investigated. It enables the adjustment of the amount of resources allocated to each individual application in order to meet its QoS requirements and tuning of its performance for some particular purposes. It can influence the outcome of packet scheduling by enabling dynamical reconfiguration of parameters and policies of packet scheduling. It can make a better resource utilization by smartly allocating unused resources to demanding applications.

In the following subsections, we are going to illustrate features of active gateways through a concrete distributed real-time multimedia application – video conferencing. We obtained a popular Internet video conferencing tool called Network Video ([Fre94]), developed at Xerox PARC. We modified the NV software to include routing facility. Active gateways are placed in the middle of video senders and video receivers so as to enforce the communications between video senders (service providers) and video receivers (service consumers) to go through the active gateways.

NV uses RTP ([SCFJ96]) as the transport protocol to transmit video data. In order to minimize the change to the NV code, we built our protocol on top of RTP, which is in turn built on top of UDP (table 2).

Table 2: Data Encapsulation in Modified Version of NV

UDP Header	RTP Header	QoS Parameters	Video Data

We have explored several adaptability features for video conferencing ([LGB95]). However, our approach towards performing QoS control was an end-to-end solution, i.e., adaptation was achieved at the end points of a video conferencing session. Moreover, QoS control decision was made based on end-point observations of network conditions and end users' requirements. End users had no control over how the video frames are routed *within* the network, nor could they perform adaptation based on the network conditions at intermediate nodes.

This is not the case any more with active gateways, however. We can perform customized control at any intermediate network nodes. Adaptation can be achieved not only at the end-points but at any intermediate nodes as well, based on their network conditions. This allows users to study a new type of adaptability that wouldn't be possible before. In the subsections that follow, we will demonstrate the functionalities of the active gateways through several experiments we conducted with our modified video conferencing tool NV.

4.2 Performance Tuner

The performance of a video conferencing session is affected by such factors as available resources (e.g. network bandwidth, buffer space, process priority), network characteristics (loss rate, latency, etc). These factors have been characterized as a set of parameters that can be set by Tcl scripts. For example,

```
setbw 200
```

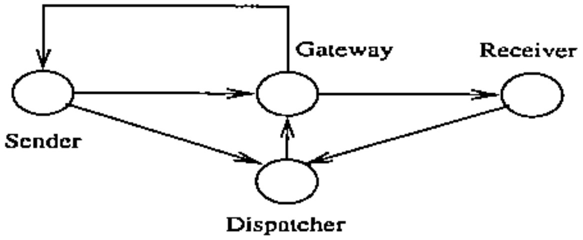

Figure 4: Performance Tuner

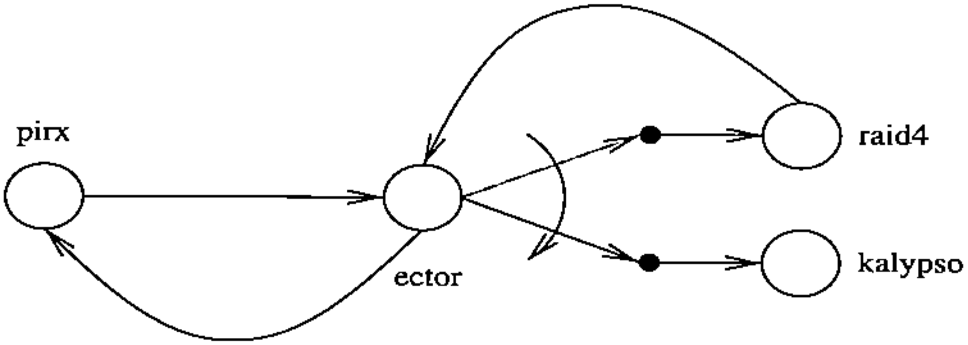

Figure 5: Redirection of Video Conferencing Traffic

sets the simulated bandwidth to 200 kbps on a node that executes this command. The change of bandwidth will cause the frame sending rate to be recomputed. Similarly, we can migrate the adaptability features from end-points to any intermediate nodes.

Figure 4 shows a simple setup of a video conferencing session. The sender and the receiver are having a one-way video conferencing session via an intermediate gateway. They periodically report their status information to the dispatcher, who, based on the feedback information, issues appropriate Tcl-scripts to the gateway for execution. Depending upon the policies defined, performance parameters at the sender side may be tuned, or certain actions are taken at the intermediate gateway to guarantee QoS perceived by the receiver, all through the execution of the Tcl-scripts at the gateway. The performance parameters that can be tunable include sender's sending rate receiver's loss rate, receiving rate, and response time. Primitives have been defined to support performance tuning operations.

4.3 Switching of Receivers

Switching of receivers of a video conferencing session is equivalent to turning off one recipient and turning on another one, which is useful in traffic control.

This experiment has the following setup (figure 5). Originally, *pirx* and *raid4* are having a video conferencing session through an intermediate router *ector*, which is also an active gateway. Now we want to set *kalypso* to be the new recipient of *pirx*. All we have to do is to issue a command, *redirect*, that tells *ector* to redirect the destination for all the frames being sent by *pirx* to *kalypso*. *raid4* is no longer a recipient of *pirx*, though it can remain as an active sender (to *pirx*). To remove *raid4* as a sender to *pirx*, command *delconn* can be used. *redirect* takes three arguments that specify for which connection the new destination will be redirected to. The Tcl script that does the switching is as follows:

```
addconn pirx raid4            /* set up connection pirx ---> raid4   */
addconn raid4 pirx            /* set up connection raid4 --> pirx    */
redirect pirx raid4 kalypso   /* do switching, now pirx ---> kalypso */
delconn raid4 pirx            /* delete connection raid4 --> pirx    */
```

Although this example seems no direct applications, it illustrates a basic functionality that can be further deployed to support more sophisticated operations at gateways. For example, it is now easy to switch the recipient among multiple parties in a round-robin or some orderly fashion. Some illusions of broadcasting a video conferencing session can be simulated in this way with proper timing control, though we have an alternate way to achieve broadcasting, which will be described below. Also, it is now easy to dynamically control the receiving of video frames based on resource availability. For example, the following code turns off some channels if its bandwidth is below a certain threshold.

```
for (i = 0; i < MAX_CHANNEL; i++) {
    if (channel[i].bandwidth < threshold) {
        reduce the data quality or shut off the channel
        available_bandwidth += channel[i].bandwidth;
    }
}
```

The bandwidths released from corresponding applications are now available to other applications. This can make the resource scheduling more flexible.

Also it is now possible to send a video session to multiple users simultaneously with personalized/customized quality of services, which will be covered in the next experiment.

4.4 Personalization of Broadcast Video Traffic

The ability to customizing systems for specific applications and/or specific end users is particularly important in multimedia applications. A typical scenario of video conferencing is one-sender and multiple-receiver (multicasting type of communication). With active network, it is very easy

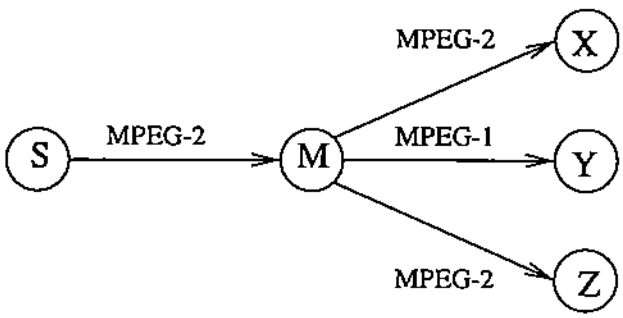

Figure 6: Personalized QoS Control

to have personalized/customized or conditional QoS control. Figure 6 shows the setup of an experiment with one sender S and three receivers X, Y, and Z.

Suppose that receivers X, Y, and Z all have their own QoS specifications. The active gateway can be configured as follows: it always guarantees delivery of video conferencing sessions to X and Z with the presentation quality X receives no worse than that Z does. It doesn't have to guarantee the delivery of video conferencing sessions to Y. So when the network bandwidth decreases, the gateway can manage to drop frames sent to Y, then degrade the quality sent to Z, and so on.

The same scenario can happen in other applications such as pay-per-view type of video-on-demand, where users willing to pay various amounts of money receive different quality of videos (HDTV, MPEG-II, and MPEG-I, etc). Although in theory customized services can be achieved for different users, a practical question arises in this scenario as to the scalability of the scheme. How do we provide customized services when the number of receivers in a network scales up?

It is certain that a hierarchical structure must be imposed on the virtual network. Figure 7 shows that many intermediate active gateways that can provide customized QoS specifications are used. They, along with the sender and receivers, form a tree structure with the sole sender being the root.

Each receiver submits its own QoS specification to its directly connected active gateway, which, upon receiving a program dispatched by the dispatcher, will execute the program to provide its QoS support. For this scheme to work correctly, however, information about QoS requirement of every receiver must be gathered and made available to every gateway whenever needed. If the directly connected gateway of a particular receiver is the only gateway that knows its QoS requirement, would the QoS always be satisfied? Maybe not, because the gateway itself may not be receiving equivalent or better QoS from the source or other gateways. Therefore, the gateway must let other gateways and eventually the source know of QoS requirements of all its directly-connected receivers.

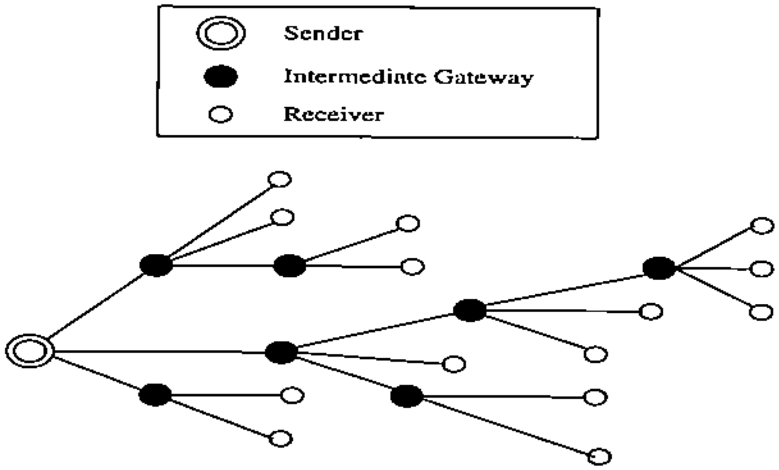

Figure 7: Personalized QoS Control

When the number of receivers is very large, propagation of the information about QoS requirements may become overwhelming. For this scheme to work efficiently, receivers with same or similar QoS requirements should be grouped together to reduce the size of the information. The grouping operations are performed from the leaves of the tree up toward the root.

Architecturally, this is similar to the way today's MBone is constructed. In fact, the RTP mixer mechanism, which is intended to be used with MBone, can also provide limited customized QoS support for different destinations. However, RTP mixers don't execute programs flowing through them; they just perform different functions (e.g. filtering and transcoding) on data flowing through them for different destinations.

4.5 Traffic Control

Probably the most important application of active gateways is in network traffic control, in which case multiple gateways may be involved. Figure 8 depicts an example scenario where A wants to talk to B. Initially only traffic from A to B exists and goes through G_1. Suppose another traffic is also going through G_1. If the total amount of traffic flows exceeds what G_1 can handle, congestion occurs. If it does nothing or limited congestion control (for example, ask one or both applications to back off), both traffic flows will be hurt. Now if the dispatcher observes this situation, it can *order* G_1 to split some of its incoming flows from A to go through G_2 by letting it executing some programs that does that. Now part of traffics from A can detour through G_2, which then delivers them to B. A more ideal scenario would be without control from dispatcher. With the absence of a dispatcher, G_1 has to figure out by itself when it likes to be congested. This can be done by observing its incoming queue length and packet average waiting time in queue. If the queue length reaches or exceeds a preset threshold, it may be about to experience

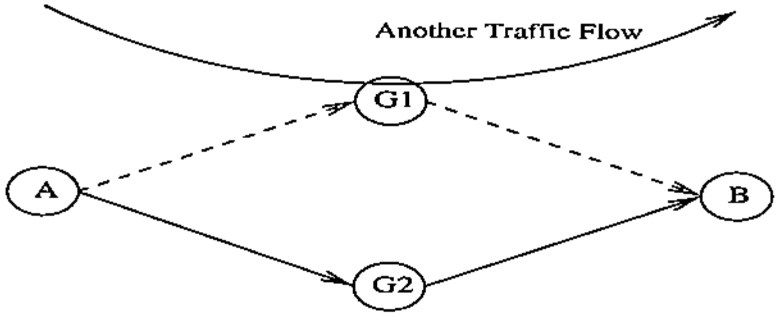

Figure 8: Detouring the Traffic

congestion. When this happens, it dispatches a program to the traffic source A to request that it send data packets with certain percentage through G_2. A, who now is an active gateway as well, updates its routing table and sends part of data packets to G_2, who will route them to the ultimate destination B. Later on when G_1 is relieved from potential congestion, it can again inform A that previous route may resume.

In this example, G_1 plays the role of the dispatcher. A is not only the sender but also an active gateway because it receives and executes Tcl scripts from the dispatcher. The condition for this to work is that all these gateways must be cooperative. For instance, during the congestion of G_1, G_2 must not be congested. And G_1 must know this information as well in order to tell A who will be the detour node.

Like conventional IP networks, our gateways change transmission paths for a traffic by modifying routing table contents. However, the way in which routing tables are modified is quite different from that in conventional IP networks. In our case, updating routing table is triggered and controlled by programs flowing around in the active network whereas in the case of conventional IP networks, modifications of routing tables occur when routers execute RIP or the like and advertise their own routes to their neighboring routers which in turn propagate the updated routes down to other routers.

Clearly, active network approach is more flexible because it can achieve finer granularity when splitting traffic. For example, traditional routing tables have one *nexthop* entry for each destination. In active gateways, each routing table can have multiple *nexthop* entries for the same destination with each entry associated with a number specifying the percentage quantity of the flow going this way.

To find out the shortest path between any two end points in the Internet, traditional IP routers rank destinations using vector-distance metric, which can be as simple as a single scalar number such as hop count. IP defines a real vector field in IP header, namely, TOS (type of service) field, which contains such measures as throughput, reliability, delay, and packet precedence ([Pos81]).

However, it had not caught much attention in practice until recently. Some new generation protocols such as IPv6 ([DH95]) have defined mechanisms for QoS-based routing support, which has been expected by emerging applications. Active gateways facilitate QoS-based routing by enabling dynamic incorporation of various policies into these protocols as well as vector-distance algorithms.

Distributed algorithms lack centralized control. State information have to be exchanged globally. Active gateways can control the degree of centralization by balancing the amounts of centralized control and decentralized control. It can marshal traffic flow among multiple active gateways. Active gateways support congestion avoidance by splitting traffic flows a priori before congestion actually occurs.

5 Discussions

5.1 Advantages of Active Gateway

One could argue that many of the services provided with active network could be implemented with traditional technology. This may be true. But one has to confess that active network provides a cleaner and more flexible and strategical way of controlling the network.

In the first section, we described the use of active technology in bandwidth allocation problem. Of course, we could embed this type of codes into traditional gateways. However, doing so has several disadvantages: first, it complicates the gateways; second, when and how to schedule resources among multiple applications is an application-level policy. It is not appropriate to incorporate application-level policies into gateways, which are supposed to provide services for all applications. Third, end users have few choices of network controls.

With active gateway, resource management and scheduling can be achieved in a broader scope. Traditionally, scheduling is typically done within the scope of a computer. Even though load balancing is typically executed among a cluster of computers, they are always tightly coupled. With active gateway, it is possible to balance the *load* or the network traffic among network nodes interconnected via several intermediate networks.

However, active network technology also introduces risks to the active nodes because they execute codes that might be from an untrust user, might be eavesdropped and masqueraded. Any practical active network must face these technical challenges.

5.2 Technical Challenges

Many technical hurdles need to be removed before active gateway can be used in practice. They include, but are not limited to, the followings.

1. **Security.** Despite its importance, this issue is not touched at all in our current prototype. How do we make sure that a program dispatched to an active gateway is from a trustable source and does not get modified along the path? Public key mechanism can be used to provide both secrecy and authenticity. For example, the digital signature of a source can be attached to each program before being sent out. However, encryption and decryption are the prices we must pay.

   Whereas eavesdropping and masquerading from malicious users can be prevented, denial of service, regardless of whether it is intentional or unintentional, has been a real challenge. A dispatched program executed by an active gateway needs to consume some resources and may require access to local file system. If the program consumes the resources unboundedly, it may affect regular local event handling and cause the local system to hang on or behave abnormally. Although we can restrict outside programs' accessibility to local resources to limit the degree of this type of attack, this action will hurt the usability and functionalities of active gateways.

   While no clear answers to this question are known, it is suggested that some policies be embedded into local resource manager. Detailed discussion of this issue is beyond the scope of this paper.

2. **Secure Communication Protocol.** If the security is not provided in network control language, it must be provided by communication protocol. When designing a secure communication protocol for a traditional network, one must choose between end-to-end encryption and link encryption. End-to-end encryption is a scheme where data are encrypted and decrypted at the source and destination only whereas link encryption encrypts and decrypts data at each node between the source and the destination. There are pros and cons for each scheme ([Den82]) and they are applicable to active network as well. The designer of a secure communication protocol for active network must decide the scheme best suited for his intended applications.

3. **Performance.** Performance is always a big issue for any system to be of practical use. Responsiveness of program execution is critical for real-time applications because the program may require immediate attention in order to have some controlled objects influenced right away. Therefore, packets containing programs for active nodes must be assigned to high priorities in order to get serviced early at every active node.

   Transmission of programs in active network must be reliable as well. Imagine that multiple programs are simultaneously dispatched to all the nodes in an active network in order to cooperatively achieve certain control function. If one program gets lost whereas others get to their destinations, these nodes cannot perform what they are expected to. Depending on what the application is, communication protocols should be properly chosen or developed to meet either responsiveness or reliability or both.

Since our active network is a virtual network built on top of an IP network using UDP, additional reliability guarantee must be provided. TCP, on the other hand, should work fine for non-realtime applications that require reliable transmissions.

Another problem associated with UDP is the limit of maximum size of a datagram, which is 64KB. For a large Tcl script, we can divide it by procedures and send them in separate datagrams because in practice a single Tcl procedure hardly exceed this limit. This way, we can avoid using TCP. To speed up runtime execution process, we can prefetch static and frequently used procedures during initialization time and load dynamic or infrequently used procedures at run-time.

4. **Distributed Object Identification.** To program the active network as a whole, there must be a mechanism in the network control language to identify distributed objects such that they can be easily referenced in programs to be executed by active nodes. Programmable naming and addressing schemes must be invented to accomplish such a task. Any logical entity, such as resource, environment, logical route, or address, is a distributed object and must be identified in one way or another.

5.3 Future Improvement

Currently, we only conducted experiments for video conferencing applications. In the future, we plan to conduct experiments for distributed video-on-demand applications and implement some transcoding algorithms at intermediate nodes. We will, through experimentation, develop and identify features of active gateways that will be useful for general applications such that they can be incorporated into operating systems' kernels in the future.

Our long term plan is to use Java as the network control language in future active gateway research. Support for Java bytecode encoding scheme is currently under way. We utilize Java's dynamic loading mechanism to achieve such a support. First of all, each Java program to be executed at an active gateway must be a valid, well-defined class. Secondly, an active gateway must invoke a Java runtime object, which, upon receiving a Java bytecode program, uses Java's dynamic class loading mechanism to verify the format of the class and resolve other classes referenced by this class. Thirdly, depending upon the system's security enforcement strategy, Java's built-in security checking mechanism can be used to ensure that the execution of the class won't violate any security constraints imposed by local security policies and cause denial of service attack. Finally, the runtime object executes the class and interacts with the router possibly implemented in platform-dependent way such as C.

Our preliminary implementation indicates that the Java bytecode encoding scheme is a promising one and that many nice features in Java are very useful in active gateways. Compared with the Tcl-scripting approach, Java's dynamic loading mechanism is very attractive and its built-in security manager facility makes security checking and policy enforcement easier. However, Java approach requires a compilation process that converts the Java source to bytecode, which is done before dispatching. Resolving classes also possibly requires multiple rounds of remote object/class fetchs, which are subject to the underlying network reliability. This process must be made efficient before the approach can become practical.

We also plan to build an interface with the World-Wide-Web to (possibly) replace Tcl/Tk. An interface with the World-Wide-Web offers great accessibility and therefore increases the usability as well. Dispatchers will be Web clients. GUIs will be implemented using Java's applet mechanism.

Researches on concurrent support of different languages will be conducted. We will study the feasibility of supporting embeddable language interpreters, language extensibility, etc. Secure communication protocols for active network will be studied extensively. We want to take advantage of the benefits active gateways offer us and develop various resource management schemes, QoS control policies, and new adaptability features for real-time distributed multimedia applications.

6 Conclusions

Active network technology promises to offer programmability, controllability, and flexibility to network infrastructure. It has great impacts on the study, development, and deployment of many important research topics. It allows new services to be supported and new applications to be generated. Our practical prototype of active gateways provides an evidence of its usefulness, though at the application level.

In this paper, we constructed a facility, called active gateway, that is intended for video conferencing traffic control. It is a virtual network built on top of conventional IP networks. We presented its architecture, design, and primary components, and demonstrated through several experiments some control functions that are not seen in conventional video conferencing tools. We also showed that this facility can be used in many aspects of multimedia communications, including resource management, real-time communication protocols, and QoS control.

Active network is an emerging technology which is still in its infancy. Our prototype did not address many important issues such as addressing/naming, security, and tunneling. It also has a lot of limitations in program encoding schemes, communication protocols for active nodes, and lack of experimentation for other applications. They leave a big room for future improvement.

References

[AMZ95] E. Amir, S. McCanne, and H. Zhang. An Application Level Video Gateway. In *ACM Multimedia'95, San Francisco*, 1995.

[Den82] Dorothy E. Denning. *Cryptography and Data Security.* Addison-Wesley, 1982.

[DH95] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 1883, Dec. 1995.

[Eid94] Hans Eidnes. MBone: The multicasting backbone. *Communications of the ACM*, 37(8):54–60, August 1994.

[Fre94] Ron Frederick. Experiences with Real-Time Software Video Compression. In *Proceedings of the Packet Video Workshop*, Portland, Oregon, September 1994.

[GM95] J. Gosling and H. McGiton. *The Java Language Environment: A White Paper.* Sun Microsystems, 1995.

[Lar96] Larry Wall and Randal L. Schwartz and Tom Christiansen. *Programming Perl, 2nd Edition.* O'Reilly, 1996.

[LGB95] Shunge Li, Shalab Goel, and Bharat Bhargava. VC Collaborator: A Mechanism for Video Conferencing Support. In *SPIE Photonics East '95 Symposium – First International Symposium on Photonics Technologies and Systems for Voice, Video, and Data Communications, SPIE Proceedings Vol. 2617*, pages 89–99, Philadelphia, Pennsylvania U.S.A, October 1995.

[Ous93] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1993.

[Pos81] Jon Postel. Internet Protocol. RFC 791, September 1981.

[SCFJ96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP, A Transport Protocol for Real-Time Applications. RFC 1889, Jan 1996.

[Sea96] J. M. Smith and et al. *SwitchWare: Accellerating Network Evolution (White Paper).* University of Pennsylvania Computer Science Department, 1996.

[TGSK96] D. L. Tennenhouse, S. J. Garland, L. Shrira, and M. F. Kaashoek. From Internet to ActiveNet. Request for Comments, Jan 1996.

[TW96] David L. Tennenhouse and David J. Wetherall. Towards an Active Network Architecture. In *Proceedings of Multimedia Computing and Networking (MMCN 96), San Jose, CA*, Jan. 1996.

[WT96] David J. Wetherall and David L. Tennenhouse. The ACTIVE IP Option. In *Proceedings of the 7th ACM SIGOPS european Workshop, Connemara, Ireland.* ACM, Sept. 1996.

[YdS96] Y. Yemini and S. da Silva. Towards programmable networks. In *IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, L'Aquila, Italy*, October, 1996.