Preface

Lin Chao
Editor
Intel Technology Journal

Today there is a host of core technologies that promises to create new and exciting uses for computing. Linking computers and computer networks so people have easy access to information—regardless of differences in computer systems, time, and geography—is key. Chief among these core technologies is increasing the capacity of multimedia—video, voice, and data communications across networks. With better and faster connections, consumers can receive complex digital content from anywhere in the world.

Multimedia networks are the focus of this Q3'99 issue of the Intel Technology Journal which has four papers. The first paper takes the important step of providing a framework to dynamically modify the functionality of the network to accommodate the requirements of the new services developed. The second paper provides an overview of the existing real and non real-time services available to Internet users. For widespread deployment of these services, there are problems that need to be solved. These problems and their solutions are also described in this paper.

The third paper focuses on telephony services developed for H.323 multimedia telephony. The architectural model for these services is different from the traditional switch model. This paper highlights how these differences affect the way services are deployed, provisioned, and charged. The fourth and final paper discusses infrastructure issues that enable real-time communication. It solves the problem of packet loss on the Internet by providing an application layer mechanism to predict and prevent the loss of audio packets, thereby improving the overall audio performance.

Multimedia Networks

By Ali Sarabi,
Director of Broadband and IP Telephony Lab, Intel Architecture Lab
Intel Corp.

During the past few years, we have all witnessed a phenomenal growth in the deployment and use of the Internet by businesses and individuals. The end-to-end network bandwidth has had to grow, and the basic data networks have had to evolve into "multimedia networks" to enable new capabilities. New types of media such as voice, video, and 3D (in addition to the traditional text and graphics) are now part of the networks and the content you and I use. As a result, new demands have been placed on the scalability, fidelity, quality of service and the overall real-time behavior of the content and the multimedia networks. A number of solutions are also under development to deliver high-speed Internet connections for the consumer. Most promising are fiber optic and "wireless fiber" links for medium to large businesses, and cable modems, xDSL and third-generation wireless for the mass market.

Intel's strategic interests and leadership in "multimedia networks" go back a few years. As early as 1994, Intel has been working on technologies, standards, and industry-enabling programs to accelerate the availability of scaleable content, quality of service, high performance multimedia and real-time communication networks and broadband access. More recently, Intel has increased its focus on "multimedia networks" silicon, consumer and business communication equipment products, and the Internet infrastructure and services to help meet Intel's "billion connected computers" vision.

One group of Internet services that is gaining considerable interest is based on real-time communication technologies and involves capabilities such as multimedia telephony. This class of applications and services pose exciting challenges to the engineering community in providing real-time performance on the Internet infrastructure that was originally designed for non-real time communications and for such services as email. In this Q3'99 issue of the *Intel Technology Journal*, authors have identified a few key problems of real-time communication applications and multimedia networks and they discuss practical solutions being implemented in Intel products and industry solutions for this market segment.

The Phoenix Framework: A Practical Architecture for Programmable Networks

Satyendra Yadav, Intel Architecture Labs, Intel Corporation
Sanjay Bakshi, Intel Architecture Labs, Intel Corporation
David Putzolu, Intel Architecture Labs, Intel Corporation
Raj Yavatkar, Communication Architecture Labs, Intel Corporation

Index words: programmable networks, multimedia, active networks

ABSTRACT

Programmable networks [1, 2] allow third parties to dynamically reprogram switches and routers in order to extend their functionality. This approach facilitates new capabilities such as dynamic reallocation of resources, automated healing from malfunctions and failures, customized information processing in network devices, and easier service creation. These capabilities enable rapid customization of the network by providing mechanisms to adapt to changing environments for new applications such as multimedia (video conferencing, video on demand, multimedia transcoders), multicast, intrusion detection, and Intranet firewalls. In this paper, we describe Intel's framework for programmable networks known as *Phoenix*. The objective of the Phoenix framework is to make it easier to deploy new network services and allow comprehensive control over the use and management of network resources and services. To enable such functionality, the Phoenix framework provides support for a mobile agent system and open interfaces. Mobile agent systems make the network very flexible as an agent can extend the network device capability on demand at run time. We also discuss how the open interfaces provided by the Phoenix framework can be utilized to deploy new network services.

INTRODUCTION

New applications such as multimedia (video conferencing, video on demand, multimedia transcoders), multicast, intrusion detection, distributed collaboration, and intranet security require that networks be capable of supporting self-reconfiguration, traffic-level monitoring, distributed and secure communication, and of adjusting to application requirements through deployment of new services [10]. Traditional network devices such as routers and switches are closed, vertically integrated systems [6]. Their functions are rigidly built into the embedded software and are typically limited to routing, congestion control, and quality of service (QoS). Traditional network architectures often have difficulty integrating new technologies and standards into the shared network infrastructure. Traditional architectures also have difficulty accommodating new services that dynamically extend the capabilities of the existing architectural model [4].

Researchers have used different approaches to overcome the limitations of traditional networks. One approach is through the definition of open interfaces [3]; that is, exposing the functionality of network devices to outside entities. Another approach, termed active networks [8, 9], uses executable programs embedded within data packets that execute on routers and switches as they traverse a network. We have adopted the former approach in the Phoenix framework by providing a set of open and safe Java*-based interfaces. In addition to open interfaces, the Phoenix framework implements a mobile agent system [5, 7] to provide a highly flexible execution environment. It also provides a safe environment so that dynamically loaded code can execute without threatening network stability or compromising data security. Moreover, the Phoenix framework preserves all the routing and forwarding semantics of current Internet architecture by

* All other brands and names are property of their respective owners.

restricting the computation environment to the application layer. This permits incremental deployment of the Phoenix framework in today's Internet.

In the next section we provide an overview of the Phoenix framework showing different building blocks and how they interact to provide a programmable environment. We describe the working of the framework using an execution scenario. We then present sample applications that utilize the Phoenix framework services to dynamically reprogram the network devices and deploy new services.

A PROGRAMMABLE NETWORK FRAMEWORK

Phoenix is an object-oriented, distributed, and security-aware programmable network framework that allows easy control and deployment of new network services. These services can be used to enable network resource management, fault diagnosis, transcoding, and new protocol support.

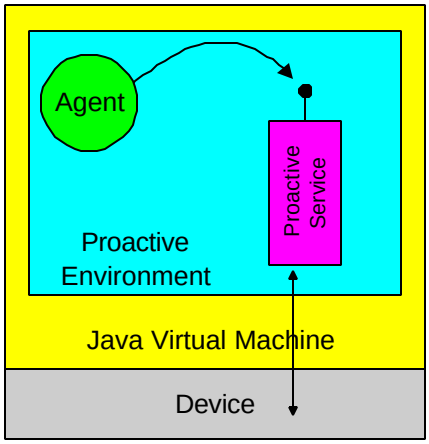


Figure 1: The Phoenix framework

Network devices that support the Phoenix architecture are called *active devices*. Legacy network devices that cannot support the Phoenix framework are called *non-active devices*. Active devices are managed from a management console called the *proactive console*. Every active device hosts a *proactive environment*, which provides basic primitives for programming and managing the device. The proactive environment is built upon a Java* Virtual machine (JVM), which makes it scalable, dynamically extensible, and secure. The JVM security features provide security support in the proactive environment. One or more *proactive services* can be installed in the proactive environment at runtime in order to dynamically extend the Phoenix framework. Proactive services are Java objects with well-defined interfaces that provide new functionality.

The Phoenix framework also defines a mobile agent system that allows an agent (an encapsulation of code and state) to be launched into the network to visit active devices and add new functionality. Such a facility makes the network infrastructure very flexible since an agent can extend the network device capability on demand.

Proactive Console

The proactive console manages the Phoenix programmable networks' framework. Network administrators can use the proactive console to install a new proactive service on active devices. The proactive console can also be used to create new mobile agents. This console is the central repository for all Java class files representing proactive services and mobile agents. In addition, it also contains the security and management policy databases that specify how the network should be managed.

Proactive Environment

The proactive environment (PEnv) is responsible for management, transport, and execution of mobile agents on active devices. PEnv is a Java object that exposes a set of well-defined Java interfaces. These interfaces are the only method available to agents for programming and managing an active device. Interfaces provided by PEnv can be invoked locally or remotely using Java Remote Method Invocation (RMI). The PEnv is also responsible for enforcing management and security policies for the active device. In order to support these policies, every PEnv is pre-installed with a number of core proactive services. These core proactive services are used to securely install and manage new proactive services.

Proactive Services

Proactive services are objects that expose one or more interfaces. Proactive services permit third parties to add a new functionality to an active device. The Phoenix framework provides the following core proactive services:

- *Basic Services* are used by agents to create, start, suspend, stop, and destroy services. Basic services also provide support for logging and thread control primitives.
- *Install and Storage Services* are invoked by the proactive console to persistently install new services on an active device.
- *Security Services* provide support for agent authentication, integrity, and authorization.
- *Communication Services* provide support for communication between agents and between an agent and other entities (e.g., a network).

- *Navigation Services* provide support for transporting an agent (with or without state) between two proactive environments.

All core proactive services are created and started when the PEnv is enabled on an active device.

A proactive service can interact with a device locally via a native library (provide access to device resources that cannot be accessed directly from Java) or remotely via Remote Procedure Call (RPC) mechanisms such as Java Remote Method Invocation (RMI), Microsoft* Distributed Component Model (DCOM), and Common Object Broker Architecture (CORBA). Legacy devices that cannot support a proactive environment can be managed via a proactive service that uses Simple Network Management Protocol (SNMP) or Command Line Interface (CLI) over a Telnet connection.

Installation

When PEnv is initially installed on a new active device, only the core proactive services are installed. New proactive services are installed and registered with the proactive environment on an active device with the help of the core proactive services. Proactive services can be installed on a device in the following three ways:

1. When the PEnv is started, it is given the address of the proactive console so that the PEnv can download the additional services from the proactive console.
2. The proactive console can install a new service on an active device by invoking install and storage services via Java RMI.
3. A service installation agent is sent to the active device that installs one or more proactive services on the device. This agent must have appropriate access rights.

Configuration

The proactive console maintains a topology database of active and legacy devices in the network. Every network device (active or legacy) is represented by a *proactive device object* (PDO). The PDO is the repository of information that is needed by PEnv to manage the device. A PDO also contains the configuration data for proactive services supported by the active device. A PEnv may contain more than one PDO when the active device is acting as a proxy for other devices (legacy, non-active). Whenever a network device needs to be managed, the proactive console sends the PDO for the device either to the device itself if it is an active device or to some other active device that acts as a proxy for managing the actual device.

MOBILE AGENT

Each mobile agent is a collection of objects. Figure 2 shows the various objects that compose a mobile agent.

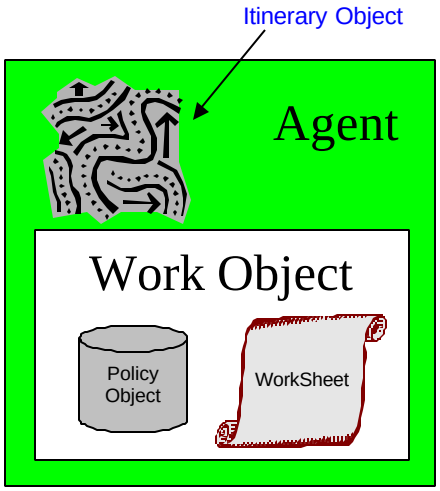


Figure 2: Mobile agent and associated objects

A mobile agent contains an itinerary object that determines active devices that the agent will visit during its lifetime. Each agent carries a WorkObject (WO) that contains the programming logic specifying how to interact with a PEnv. A WorkObject does not have any relationship to the itinerary object, which is only associated with the agent. Associated with the WO is the policy object that defines what the agent is to do when erroneous events occur. For example, the policy will specify the action to be taken when the agent visits a device, but cannot execute because it cannot obtain the necessary resources for execution. Each WO contains one or more WorkSheet (WS) objects that specify what the agent does at each device it visits and also includes any policies that indicate how the WS should be executed. For example, a policy may specify the conditions that determine whether or not the WS should be executed at each of the devices the agent visits during its itinerary. Another policy may specify the services that are required to execute a WS so that an agent can help the PEnv determine the resources needed. Figure 2 shows an agent with a single WorkSheet.

*The Phoenix framework supports a variety of WS objects. A WS may contain a Java object that invokes one or more proactive services or just a simple Java script that gets executed at a device by the JavaScript proactive service. This allows for a wide variety of agents to be authored and deployed.

Figure 3 shows how an agent interacts with various services inside a PEnv. When an agent arrives at an active device, it gets authenticated. Once the agent has

been validated and passed all the security checks, its WorkObject is allowed to execute. Figure 3 also shows two types of proactive services. The proactive service on the left executes inside the PEnv, whereas the proactive service on the right interacts with some native library to manipulate the actual device.

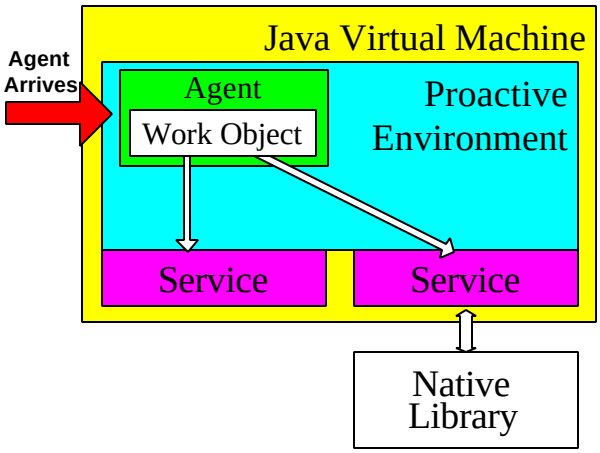


Figure 3: Agent execution

Security Enforcement

The PDO defines the security access level (SAL) needed to access each associated proactive service. Any agent that needs to access a proactive service must be digitally signed by a principal that has an SAL that is at least equal to or higher than that defined in the PDO for that proactive service. The PEnv reserves the right to reject access to the requested proactive service even if an agent has a sufficient SAL. This can occur when the device under the control of the PEnv is running low on some critical resource such as CPU or memory. An execution priority level is associated with each agent that allows execution of higher priority agents first.

Typical Interactions at an Active Device

Figure 4 shows what happens when a mobile agent arrives at an active device. The figure shows how agents, PDOs, and proactive services execute under the control of the PEnv.

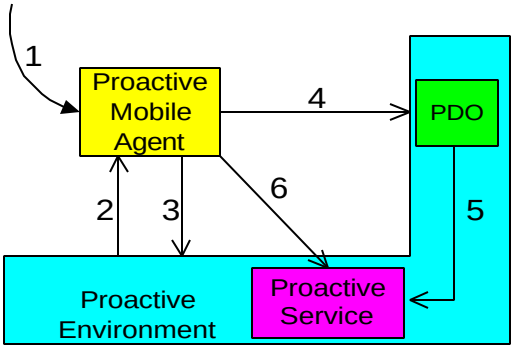


Figure 4: Component interaction

Following are the high-level actions that are taken upon the arrival of a mobile agent:

1. A mobile agent arrives at the active device. At this point it is not executing.
2. The PEnv verifies the security access level and execution priority level for the agent to decide if the agent should be allowed to execute. If the agent has enough access rights and the necessary resources on the device are available then the control is passed to the agent.
3. The agent first makes a request to the PEnv for the interface of the PDO for the device it wants to operate upon. If the requested PDO is not available, the agent terminates or may move to another device. This is governed by the policy object associated with the agent WorkObject.
4. Assuming that the PDO for the device is available, the agent then asks the PDO for the proactive service it needs access to.
5. The PDO creates and configures an instance of the requested proactive service and returns a reference to the agent.
6. The agent invokes the proactive service to perform the necessary actions.

USE OF JAVA TECHNOLOGY

The Phoenix framework uses Java* technology for several reasons. Java is an object-oriented language that inherently supports security. Java was also developed with software mobility and platform independence in mind. Both of these features are critical for programmable networks for the following reasons:

- Networks are composed of heterogeneous devices and developing a solution that works across these devices is important.
- Network devices have limited resources and are not typically designed to support or envision future requirements. The ability to dynamically add new functionality through runtime installation is important to provide extensibility.

APPLICATIONS OF THE PHOENIX FRAMEWORK

The Phoenix framework is a highly flexible, extensible architecture, making it a powerful foundation for adding new services and capabilities to computer networks. This section describes three different application areas where

the Phoenix framework is useful for enhancing networking functionality.

Scriptable Remote Network Management

Many tasks necessary for the management and ongoing maintenance of computer networks can be performed by automated scripts. Traditionally, such scripts are executed at an administrator's console, using such tools as Command Line Interface (CLI) or SNMP via the network to manipulate the state of routers and switches. Using the Phoenix framework, a prototype was constructed that performs some well-understood network monitoring tasks (link utilization monitoring and configuration) in an automated fashion.

The prototype consists of a testbed network containing several routers and switches and several PCs, some acting as endpoints and some acting as network devices. Each PC contains a proactive environment, making it an active device. These active devices are used by agents to perform a variety of tasks, including network monitoring (by examining loads on various links), network configuration (via a proxy SNMP service), and endpoint control (via a service for launching and controlling applications on a PC). This prototype has been used to demonstrate the benefits that the Phoenix framework brings to network management. The most notable benefit in this area is the ability to execute scripts from any point in the network including on active devices connected to routers and switches via highly reliable paths such as single hop Ethernet links or serial ports. The Phoenix framework can do this because it uses mobile agents, which can execute their WorkSheets (in this case, scripts) at any point in the network where a PEnv is present, rather than being bound to an administrator's console. Using this approach, scripts are able to perform their duties even when network conditions (such as link over utilization) make it otherwise impossible for a network administrator to access a network node. Another example where the execution of scripts at any point in the network is shown to be important is when determining connectivity. For example, the *ping* program can be used to determine connectivity between two hosts located at the edges of a network. Agents are able to invoke ping via scripts executing on devices in the middle of the network; allowing the connectivity between any two points in the network to be verified from anywhere in the network.

A further advantage of the Phoenix framework in this application domain is that scripts are written in a much more generic fashion than they used to be. In the prototype constructed, a script would periodically examine the data rate of a link, and based on whether the link was becoming saturated, cause some of the traffic to take alternate paths. Rather than writing the script directly against the particular mechanism (e.g., SNMP or CLI) used to control individual devices, the script uses the higher level constructs presented by proactive services. These constructs allow the script to simply perform function calls to retrieve current utilization and set link status, ignoring both the actual method used to get/set this information (leveraging the ability of services to hide implementation details) and any specific configuration or access control information that is associated with particular devices (via the capabilities provided by PDOs).

DYNAMIC APPLICATION DEPLOYMENT

The services provided by computer networks can be roughly divided into two categories. One category consists of devices that are based on highly specialized, typically fixed function equipment, examples of which include switches and routers. The second category of devices much more closely resembles traditional PCs where software is used to provide network services. Examples of the second category of devices include Web caches, firewalls, media gateways, and H.323 gatekeepers. This second category of devices is a potentially useful area to apply the Phoenix framework, as described below.

Certain kinds of network value-added applications described in the second category above are in constant use, for example, firewalls and Web caches. Other applications however tend to be used relatively infrequently such as a multimedia transcoding gateway (which might only be needed during an online corporate presentation such as a shareholder's meeting) or packet sniffing functionality (used only when diagnosing network problems or attacks). While firewalls and Web caches may be deployed in a relatively permanent manner to good effect, it may not be cost effective to dedicate individual computing resources to rarely used applications such as a transcoder or sniffer.

The Phoenix framework provides a powerful solution to the problem of infrequently used network applications. Rather than dedicating some device or devices to each application, value added applications can be deployed as services or as agents, depending on the performance and flexibility needed. For compute intensive applications, services are the preferred approach because they may consist almost entirely of native code, which is often preferable to Java bytecodes for performance reasons. Applications that do not require significant computation but are best deployed in a more mobile fashion (e.g., on demand deployment of signaling gateway on a device that is currently unused) may be better served by agents due to their ability to migrate on demand.

Intrusion Detection

With the ever-increasing reliance of corporations on both the internet and corporate intranets for essential business activities, protecting the network infrastructure from security attacks has become an important problem. One class of attacks involves an intrusion whereby a knowledgeable individual compromises security by introducing traffic in a network that often appears to be perfectly legitimate, yet can cause a variety of problems and malfunctions in the network and connected hosts. Such intrusions are hard to detect, monitor, or trace back to the culprit from a single vantage point in the network. Furthermore, intruders may masquerade as another account or another machine than the one they are actually sending from (via techniques such as IP spoofing) thereby making it very difficult to deal with such intrusions using traditional approaches. In order to combat such intrusions, a cooperative effort by many network devices and access to the distributed state (state information that captures the operational state of many dispersed network devices scattered in a network) are necessary.

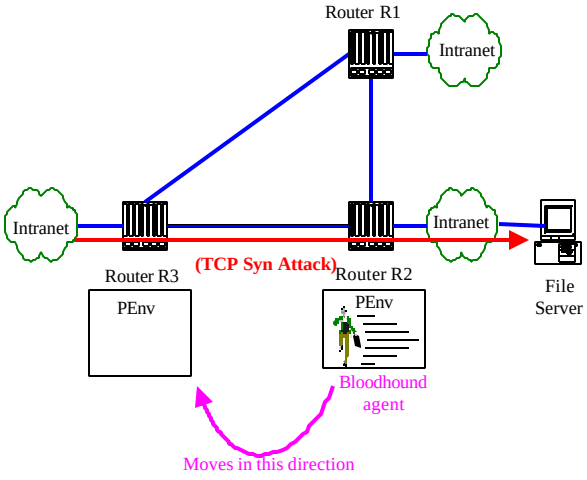


Figure 5: Intrusion detection

A well-known example of a hard-to-combat network attack is the so-called "TCP SYN attack," where an attacker floods a target machine with an incessant stream of TCP SYN packets. This causes the TCP software of the target machine to be overcome with wasteful busywork attempting to respond to the SYN packets by continually attempting to establish new TCP connections. This can essentially make the target machine unavailable for bona fide TCP connections.

The Phoenix framework could be used to allow the network administrator to "trace" the source of such an attack, even in the face of IP spoofing or other methods of obscuring the source of an attack. In the proposed design, the Phoenix framework approach to combating these attacks would consist of two agents: a "watchdog" agent, configured to observe the reachability of servers that might be subject to attacks and a "bloodhound" agent, used to trace an attack back to its source. Also necessary are four services: a connection monitoring service, a flow interception service, a network topology service, and a sniffer service.

The watchdog agent uses the connection monitoring service to periodically establish connections with a specified set of servers. If the agent is unable to establish a connection for a user-defined interval, it raises an alert and launches a bloodhound agent. The bloodhound agent is configured with the address of the server being attacked and sent to the active device that is "closest" to the attacked server. Once the bloodhound agent arrives at the active device, it uses the flow interception service to instruct the routers closest to the server to temporarily redirect traffic destined for the server to the agent, which uses the sniffer service to examine the redirected traffic. Using fairly simple statistical techniques, the bloodhound agent determines which router is carrying the majority of the attack traffic. Given this information, the bloodhound agent then moves itself again, this time to the active device closest to the router in question, where "closest" is determined using a network topology discovery service, which in turn could utilize well known repositories such as HP OpenView or Active Directories. The bloodhound agent then repeats its previous behavior, examining each of the routers connected to the specified router for attack traffic. Using this technique, the bloodhound agent is able to eventually find the attack ingress point in the network.

The Phoenix provides a highly automated solution to this problem that would otherwise require an administrator to manually "sniff" multiple links in a network, typically requiring carrying a sniffer around and analyzing captured traffic. Using the mobility of agents and the abstractions provided by services, a much more powerful, easy-to-use approach is possible.

CONCLUSION

In this paper we presented a framework for programmable networks, called Phoenix, that provides an open interface for third parties to program network devices. The Phoenix framework also provides a mobile agent system that makes the network infrastructure much more flexible as a mobile agent can extend network device capability on demand. Other interesting uses of mobile agents are their ability to traverse the network performing inter-dependent tasks. The Phoenix framework provides a safe environment for

dynamically loaded code to execute without threat to network stability and data security.

The Phoenix framework provides programmability features that can be used in a range of practical applications spanning from scripts to dynamically loaded mobile agents. We also demonstrate applications of the Phoenix framework by means of three application scenarios. The Phoenix framework provides an extensible framework that allows programmers and network administrators to extend device capabilities and deploy new services to meet the new and evolving requirements placed on computer networks.

REFERENCES

[1] CANES: Composable Active Networks Elements, http://www.cc.gatech.edu/projects/canes/

[2] NetScript: A Language and Environment for Programmable Networks, http://www.cs.columbia.edu/dcc/netscript/

[3] Biswas, J. et.al. "The IEEE P1520 Standards Initiative for Programmable Network Interfaces," *IEEE Communication Magazine*, October 1998, pp. 64-70.

[4] Decasper, D. et.al. "Router Plugins – A Modular and Extensible Software Framework for Modern High Performance Integrated Services Routers," *Proceedings of ACM SIGCOMM'98*, September 1998.

[5] Hartman, J. et al., "Joust: A Platform for Liquid Software," *IEEE Computer Magazine,* April 1999, pp. 50-56.

[6] Peterson, L. et. al. "OS Support for General-Purpose Routers," *HotOS Workshop*, March 1999.

[7] Pham, V. and Karmouch, A., "Mobile Software Agents: An Overview Management," *IEEE Communications Magazine,* July 1998, pp. 26-37.

[8] Smith, J. et.al., "Activating Networks: A Progress Report," *IEEE Computer,* April 1999, pp. 32-41.

[9] Psounis, K., "Active Networks: Applications, Security, Safety and Architectures," *IEEE Communications Survey*, First Quarter, 1999. http://www.comsoc.org/pubs/surveys/

[10] Wetherall, D. et.al., "Introducing New Internet Services: Why and How," *IEEE Network,* May/June 1998, pp. 12-19.

AUTHORS' BIOGRAPHIES

Satyendra Yadav is a software engineering manager at Intel Architecture Labs where he leads research and development of programmable networks and next-generation switching and routing architectures. Satyendra's other research interests are policy-based network management, quality of service in the Internet, and network security. His e-mail is Satyendra.Yadav@intel.com.

Sanjay Bakshi is a senior software engineer at Intel Architecture Labs. His primary areas of interest are mobile agent architectures, object-oriented programming, and architectures used for routers and switches. His e-mail is Sanjay.Bakshi@intel.com.

David Putzolu is an architect at Intel Architecture Labs. David's research interests are in the areas of open networking architectures, active networks, multimedia streaming, adaptivity, and quality of service. His e-mail is David.Putzolu@intel.com.

Raj Yavatkar is currently the Director of Internet Infrastructure at Intel's Communication Architecture Laboratory where he leads Intel's efforts in the areas of active networks, policy-based network management, and end-to-end Quality of Service (QoS). At Intel, Raj has played a major role in defining Intel's Internet QoS roadmap and has led the development of RSVP and a policy-based network management framework. Raj was also a principal contributor to the design of the packet scheduling framework and Generic QoS API for the next generation of operating systems from Microsoft. His e-mail is Raj.Yavatkar@intel.com.

Introduction to Personal Communication on the Internet

Christian Dreke, Communication Architecture Lab, Intel Corporation

Index words: personal communication, buddy list, Internet presence, non real-time and real-time communication

ABSTRACT

This paper focuses on the burgeoning field of Internet Personal Communication Channels/Services. Even though e-mail has been with us for a long time, it is only now with the increasing acceptance of the Internet by a broader spectrum of the population that the true potential of the Internet as a personal communication medium is starting to be explored. The paper discusses a number of existing personal communication channels/services and the applications that utilize them.

Real-time communication channels such as instant messaging, text chat and audio/video streaming and non real-time channels such as e-mail, newsgroups and shared spaces are described with respect to their use in personal communication. Furthermore, personal communication applications such as buddy lists, virtual communities, and others are discussed. Also mentioned are the common problems faced by these applications. And finally, a standards-based solution to solve existing interoperability problems is called for.

More specifically, a class of applications known as buddy lists is discussed. Buddy lists are currently the most popular communication application for real-time personal communication over the Internet. Today's buddy lists do not interoperate. They use proprietary technology to detect when users go online and real-time communication between users is possible. Hence, communication is only possible when all communicating parties use the same buddy list client software; that is, the same fixed and proprietary set of communication channels (e.g., instant messaging, text chat, etc.) is used.

In order to realize a greater degree of innovation in the field of Internet personal communication and to make it easier for existing and new communication channels to gain acceptance and be used ubiquitously, it is necessary to develop a standards-based framework for Internet-based personal communication. Standards necessary to achieve full interoperability of communication applications and tools include Internet-scale standards for publishing, finding, and activating personal communication channels or services, as well as standards for the communication channels themselves.

INTRODUCTION

The Internet marks a dramatic shift in how people acquire information. Much of the success of the Internet has been due to the fact that it provides up-to-date information at your fingertips. However, information access is only part of the Internet's potential. Electronic commerce or e-commerce, business-to-business communication, and personal communication are all areas where the Internet is gaining acceptance and is transforming our society.

Today more and more people are using the Internet for personal communication. In this paper, a number of real-time and non-real-time communication channels for personal communication over the Internet are presented. Existing classes of applications incorporating personal communication channels are also discussed.

PERSONAL COMMUNICATION CHANNELS

All communication channels described below use the Internet as the underlying communication medium and are suitable for Internet-based personal communication, i.e., personal human-to-human interaction via the Internet.

Real-Time Channels

Real-time communication channels are channels in which communication is highly interactive, and all of the communicating parties are simultaneously present.

Instant Messaging

Instant messaging is a relatively new communication channel. It has been popularized by buddy lists, which are described later. Instant messaging allows for the sending of messages from one party to one or more other parties. Currently, instant messages consist of short text messages, although one could imagine an instant message

containing sound, graphics, and other types of media.

Instant messaging and e-mail differ primarily in their real-time nature and delivery mechanisms. Whereas e-mail uses a potentially lengthy store-and-forward delivery mechanism, an instant message is delivered immediately from sender to recipient. Hence, using instant messaging, the sender knows that messages will be delivered to the receiver almost "instantaneously." Furthermore, due to the fact that instant messages tend to be very short and unobtrusive, they usually demand little attention from the recipients.

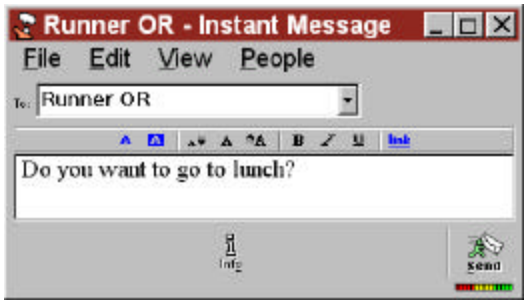


Figure 1: Composing an Instant Message using the AOL Instant Messenger*

Figure 1 shows an example of composing an instant message using the AOL Instant Messenger BuddyList*. Currently, no standard exists for sending and receiving instant messages. However, the Instant Message and Presence Protocol IETF working group is working toward a standard for both instant messaging and for detecting the online presence of users.

Text Chat

Text chat is very similar to instant messaging, except it is better suited to a higher degree of interactivity. Usually the text chat client application transmits every character typed in real-time or a complete message that is terminated by the user pressing the *enter* key. Even though the separation between text chat and instant messaging can become blurred, as both are used to exchange short text messages interactively between two or more parties, text chat is intended for two-way communication and tends to show a higher degree of interactivity and involvement between its users. Some instant messaging programs such as the one offered by PeopleLink allow for conversion of an instant message into a two-way text chat.

* All other brands are the property of their respective owners.

Figure 2: Text Chat client

Figure 2 shows PeopleLink's Text Chat* applet. There is currently no standard for small group text chat, as employed by most buddy lists. However, there is an IETF standard for establishing topical chat for large groups of people. This standard is known as Internet Relay Chat (IRC) [6]. IRC uses chat rooms, which are "chat areas" for people to meet and discuss topical issues related to the overall theme set by the room. IRC may be thought of as the real-time equivalent of newsgroups. Many IRC-compliant chat clients are in existence today.

Internet Phones (Audio/Video Streaming)

Internet phones support real-time communication over the Internet by sending and receiving audio and video streams. They are able to provide a very rich and engaging real-time communication channel.

A set of standards for call setup and control, as well as audio and video compression and streaming, is described within the H.323 standard as approved by the International Telecommunications Union (ITU) in 1996. A number of H.323-compliant Internet phones are available today, including Microsoft NetMeeting* and the Intel® Internet Phone.

Although a description of the H.323 standard is not within the scope of this paper, it should be noted that H.323 is more than just a standard for Internet phones. It aims at being the standard framework for most, if not all, real-time communication including video, audio, and data sharing over the Internet.

Whiteboard

A whiteboard allows communicating parties to share textual and graphical information in real time. It is typically used in the context of some other primary communication channel such as an audio/video phone call or a presentation. The whiteboard communication channel imposes little or no structure on the communication other than the context or theme provided externally. Whiteboards have not been used heavily for personal communication mainly because they often rely on a

primary communication channel to be established first. Furthermore, the user interfaces of current whiteboards have not been geared towards small group personal communication; rather, they have been geared towards professional data-sharing activities.

Within the T.120 protocol family that is part of the H.323 standard, the ITU has defined an application layer protocol for the real-time exchange of screen data and annotations called the Multipoint Still Image and Annotation Protocol (T.126). This protocol provides a standard way for whiteboards to exchange information in real time.

Comparison of Real-Time Communication Channels

Real-time communication channels differ in their bandwidth requirements and user engagement requirements.

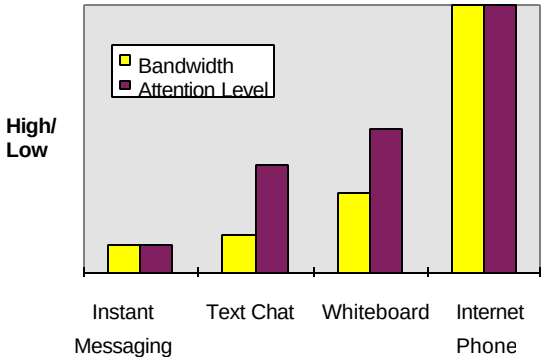


Figure 3: Relative bandwidth and level of engagement/attention requirements for the different communication channels

Figure 3 highlights the major differences between the different types of real-time personal communication channels. One of the major differences is the bandwidth requirements for a particular communication channel. Instant messaging is in part very popular because of its low bandwidth requirements. It allows for other communication or PC activities to occur with negligible impact on performance. Internet phones, on the other hand, require more bandwidth but they provide a much richer and more engaging personal communication channel.

The level of engagement between the communicating parties also varies significantly between the different communication channels. Even though instant messaging is considered a real-time channel, it is typically done as a background task. On the other hand, text chat is geared towards two-way communication with a slightly higher degree of engagement between the communicating parties. Finally, Internet phones provide a rich, highly engaging, communication channel most suitable to be the primary channel at the time of use.

Non Real-Time Channels

Non real-time personal communication channels, as discussed below, are channels in which the message delivery requirements are relaxed. Messages may still be considered useful even if they take days to reach the recipient(s).

The asynchronous nature of non real-time communication channels accommodates differences in the availability of users to communicate. Any messages can be consumed at the reader's leisure, rather than the convenience of the sender. Furthermore, since most non real-time communication channels, including all those discussed here, require an always-connected server of some sort, message archiving and the mobility of the communicating parties can be easily supported.

E-mail

E-mail is still the most widely used Internet communication tool for business as well as personal communication. The standard for e-mail message exchange known as the Simple Mail Transfer Protocol was written by John Postel in 1982 [1]. E-mail is considered non real-time because the sender and recipient do not need to be simultaneously present to communicate. Instead, e-mail utilizes a store and forward architecture where messages are passed from mail server to mail server and finally to the recipient with no strict bounds on delivery time. The biggest strength of e-mail is the fact that it has been so widely adopted. Also, many tools such as e-mail filters and mailing lists are available that allow users to customize this communication channel.

Newsgroups

Like e-mail, newsgroups have been around for a long time. In fact, long before the Internet became mainstream, USENET newsgroups were primarily used by many people to exchange technical information.

The format of newsgroup messages and the transport were standardized in 1983 by the "Standard for Interchange of USENET Messages" [2] and in 1986 by the "Network News Transfer Protocol" [3], respectively.

Like e-mail, newsgroups enjoy tremendous widespread usage. However, as an intimate small group communication channel, USENET newsgroups have become less useful. This is partly due to their growing success, which has resulted in a lot of posting of unrelated messages (i.e., spamming), but also due to their lack of access controls. Without access controls, newsgroups are best suited for non real-time

communication between medium to large groups of people.

Shared Spaces

Non real-time shared spaces are the persistent counterparts of real-time shared spaces such as whiteboards. For example, a family Web page might be considered a primitive shared space for small group personal communication. A better example is a persistent whiteboard with access controls allowing only a well defined group of people to view/modify the contents of the whiteboard. This whiteboard is analogous to a refrigerator door with Post-it* notes.

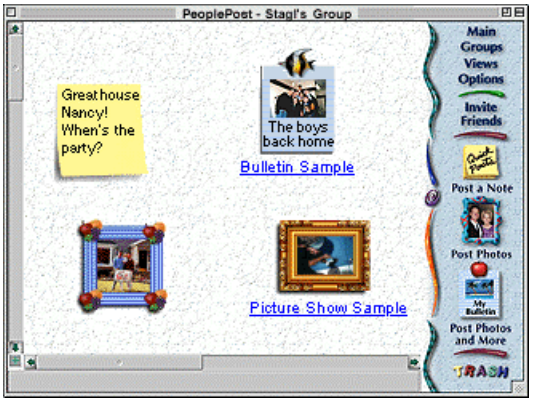


Figure 4: Secure small group shared space example

An example of a shared space is PeoplePost* (http://www.peoplepost.com). Communication within the shared space centers on one theme, such as the life of a particular family.

No standard for exchanging information in small group shared spaces exists. However, a related standard for distributed authoring of documents has just reached RFC status. The "HTTP Extensions for Distributed Authoring" or WebDAV [4] standard describes a set of HTTP extension headers to be used for distributed authoring of documents. WebDAV may be used as the foundation for an application layer protocol to define a standard for small group shared spaces.

Comparison of Non Real-Time Communication Channels

One way to compare non real-time communication channels is by their degree of generality. The degree of generality is determined by the amount of pre-determined structure and context built into the communication channel. The more structure that is provided by the channel the less general it becomes.

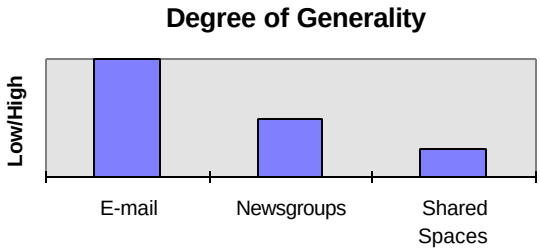


Figure 5: Relative degree of generality of non-real time communication channels

As illustrated in Figure 5, e-mail is the most general of the illustrated non real-time communication channels, since it makes no assumptions as to the topic or theme of the communication. Newsgroups typically limit themselves to specific topics or themes and it is up to the communicating parties to provide structure to the conversation. Finally, shared spaces usually have a theme to guide the communication and often impose structure on the type of information that is to be exchanged.

APPLICATIONS

All communication occurs with a certain amount of context. This context may be provided through pre-existing personal relationships, through common interests, or simply by being in the same place at the same time. For example, buddy lists, which are described next, are most useful when the communicating parties have already established a relationship by some other external means.

Buddy Lists

In order to understand buddy lists, it is important to first understand what problem they are helping to solve.

When the Internet transitioned from an academic network consisting of computer networks in universities and government agencies to include millions of computers in people's homes, it started to become a viable platform for real-time personal communication. Like many technological advances, buddy lists were born out of a need to solve a particular problem, namely, facilitating person-to-person real-time communication in a world of transiently connected users.

The main problem that buddy lists are helping to solve is being able to detect when users log on to the Internet and become available for real-time communication. This problem is compounded by the fact that the addresses used to identify hosts on the Internet, namely Internet Protocol Version 4 (IPv4) addresses, are in short supply.

Due to the enormous growth of the Internet, the supply of unassigned IPv4 addresses has been shrinking

dramatically. Therefore, most Internet service providers (ISPs) implement a scheme in which they can make efficient use of their IP address pool.

Many users connect to the Internet through dialup connections and are dynamically assigned an IP address out of the ISP's address pool for the duration of the dialup connection. After a user disconnects, their IP address may be re-used for another user that subsequently connects to the ISP. This scheme allows an ISP to support a large number of transiently connected dialup users with a smaller pool of IP addresses.

Unfortunately, this means many users get assigned different IP addresses every time they dial up to their ISPs and log on to the Internet. Furthermore, since IP addresses are needed to find endpoints on the Internet, it makes it difficult for users to find each other for real-time communication over the Internet. Buddy lists solve this problem by providing a dynamic directory that maps a user's unique static identifier unique within the particular directory service to the IP address currently in use by the user. In addition to keeping track of users assigned IP addresses, buddy lists may also keep track of other pertinent information such as when the user is busy or away from his/her PC.

Hence, one of the most significant features of buddy lists is their ability to detect the Internet presence and willingness/ability of users to communicate, as well as to determine the IP addresses currently assigned to those users. This enables them to become facilitators of real-time communication, even when the communicating parties are only transiently connected to the Internet. Most buddy lists provide users with the choice between a number of real-time and non real-time communication channels. Instant messaging and text chat are commonly used.

Figure 6: Tribal Voice's PowWow buddy list

Figure 6 shows PowWow*, a buddy list developed by Tribal Voice (http://www.tribal.com). Other examples of buddy lists include Mirabilis' ICQ* (http://www.mirabilis.com), AOL's Instant Messenger* (http://www.aol.com), Activerse's Ding* (http://www.activerse.com), and the Yahoo! Messenger* (http://www.yahoo.com). Some of these buddy lists employ highly scalable peer-to-peer presence detection solutions, such as Activerse's Ding. Others, such as AOL's Instant Messenger, have opted for a more centralized solution for routing presence and content information. This gives them centralized logging capabilities and control to enforce access policies. However, all buddy lists solve the problem of indicating when a buddy is logged onto the Internet and therefore available to communicate.

Finally, some applications also embed a presence technology solution, as employed by buddy lists. For example, Fashion>>Trip* by ModaCAD is an application targeted at teenage girls that allows two friends to shop for clothes in a virtual shopping mall and to do so collaboratively (e.g., while talking on a video phone). Fashion>>Trip embeds a presence solution that allows for two or more users to detect each other's online presence, and it enables them to communicate and collaborate while shopping for clothes in the virtual mall.

Internet Phones

Internet phones such as Microsoft NetMeeting* and the Intel® Internet Video Phone are standalone applications that allow real-time communication over the Internet between two parties using audio, video, and data streams. Since Internet phone applications incorporate real-time communication channel(s), they must, just like buddy lists, deal with the problem of detecting when users are logged on to the Internet and how they may be reached.

To address this issue, Microsoft's NetMeeting* and the Intel Internet Video Phone provide a dynamic directory service based on Microsoft's Internet Location Service. In the future, Internet phone applications may benefit from a standard for the detection of Internet presence.

As mentioned previously, the communication channels provided by H.323-compliant Internet phones have already been standardized through the International Telecommunications Union's H.323 standard. However, finding the endpoints that one wants to communicate with is outside the scope of the H.323 standard and has yet to be addressed.

*All other brands are the property of their respective owners.

Virtual Communities

While buddy lists and Internet phone applications for personal use focus on people who want to communicate in real-time with a small set of people that they know, virtual communities are often organized around interests and topics. In fact, virtual communities are providing meeting grounds on the Internet for establishing and fostering relationships.

Virtual communities often make use of real-time communication channels such as text chat or multi-party audio chat, as well as non real-time communication channels such as message boards or newsgroups. The Well (http://www.well.com) is an example of a prominent virtual community. It was started in 1985 as a BBS based community in New York city and transitioned to the Internet in 1992. It has become an early success story for virtual communities.

Other Internet-based virtual communities include Web sites such as Geocities (www.geocities.com), Tripod (www.Tripod.com), and iVillage (www.iVillage.com); game aggregation sites such as the World Opponent Network (www.won.net) and Mplayer (www.mplayer.com); and audio chat communities such as hearme.com and Talkcity (www.talkcity.com).

Virtual Shared Spaces

Virtual shared spaces allow for private sharing of information between a small group of people that want to communicate within a particular context or theme. The context could be based on relationships such as a family or extended family, on interests or hobbies, or on specific tasks such as planning and/or collaborating on a particular project.

Virtual shared spaces are a relatively new genre on the Internet. Examples of virtual shared spaces include FamilyShoebox (http://www.familyshoebox.com), PeoplePost (http://www.peoplepost.com), eCircles (www.eCircles.com), Visto (www.visto.com), and eGroups (www.egroups.com).

Although virtual shared spaces may incorporate some real-time elements or communication channels, they do not tend to focus on real-time communications. Instead, they typically provide a shared space for storing and viewing messages, files, pictures, drawings, and other multimedia content. Most also offer individual and/or shared calendar scheduling and address books.

Some virtual shared spaces provide an overall theme or context for group communications. For example, for FamilyShoebox the theme is communication within a family or extended family. Other virtual shared spaces such as Visto allow the context to be set by the group and can support task-based activities including task-related planning, scheduling, and communication.

Multi-User Virtual Worlds

Multi-user virtual worlds are two- or three-dimensional graphical environments that allow people to meet and communicate in a graphical virtual space. They are often referred to as MOOs. A MOO is an Object-Oriented version of a text-based multi-user virtual world, commonly referred to as Multi-user Dungeon or MUD. In a MOO, users can typically define their personal two- or three-dimensional representations called avatars. Users interact and communicate with each other through their avatars.

Current multi-user virtual worlds employ real-time communication channels such as multi-user audio chat, text chat, and other real-time channels to allow for communication between users. Examples of multi-user virtual worlds include the Palace (www.thepalace.com) and the OnLive Traveler (www.onlive.com).

The virtual reality modeling language (VRML) defines a standard file format for representing three-dimensional multimedia contents. In particular, a VRML file describes a graphical space containing three-dimensional and/or aural objects and ways to interact with them.

VRML version 1.0 was based on the Open Inventor* file format from Silicon Graphics. In December 1997, VRML97 became an international standard (ISO/IEC 14772) for describing interactive 3D multimedia on the Internet. VRML97 is based on VRML 2.0. More information about VRML is available at www.vrml.org.

CURRENT BARRIERS

One of the biggest barriers for personal communication over the Internet today is the fact that many of the standards necessary to ensure compatibility between the various communication applications are missing. For example, buddy list applications use proprietary presence detection protocols. The Instant Messaging and Presence Protocol (IMPP) IETF working group is trying to solve this problem by working towards a standard for presence detection and instant messaging.

However, creating a standard for detecting a user's Internet presence alone is not sufficient. In order for buddy lists and other communication applications to be truly interoperable, they need to be able to determine in a standard way what communication channels are supported and available between the parties that wish to communicate. Hence, there is a need for a standard for publishing the availability of real and non real-time personal communication channels. Such a standard is necessary to allow communication applications from

different vendors to establish common communication channels.

The communication channels themselves may or may not have agreed-upon standards for information exchange. Widely used channels such as instant messaging would benefit from a standard that allows for the creation of multiple interoperable clients. For other channels, such as those employed by most multi-player games, there may be little or no incentive to create multiple interoperable clients. Hence, they need not be standardized.

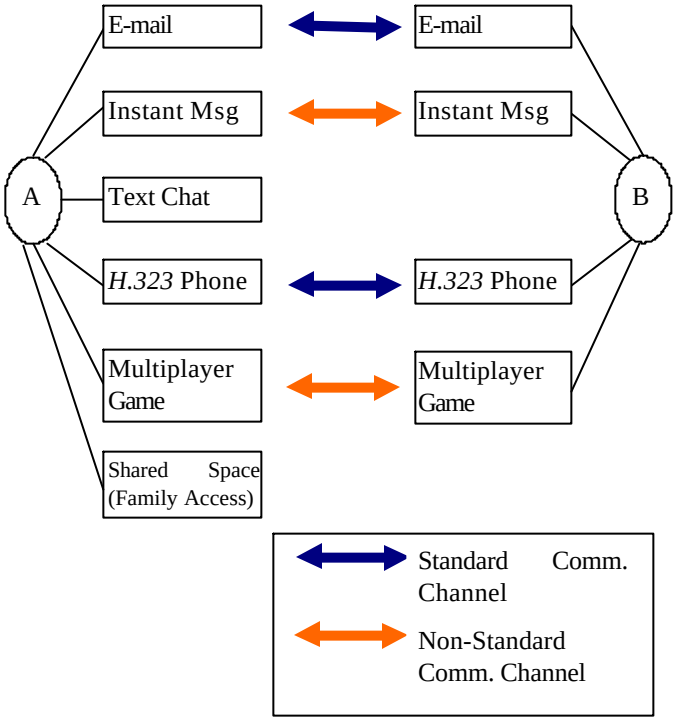


Figure 7: User A and user B determine that they have four communication channels in common

Figure 7 demonstrates user A and user B dynamically matching on four common communication channels. Having detected a common match, user A and user B may then proceed to choose the appropriate channel(s) from the common set, i.e., e-mail, instant messaging, H.323 Internet phone, and/or a particular multiplayer game, to satisfy their communication needs.

In order to facilitate interoperability of various personal communication applications and tools, the following standards are needed:

1. *Detection of real-time and non real-time communication channels and their availability for a particular person, group of people, or endpoint(s).* This involves creating a standard for publishing communication channels and their current availability status, as well as being able to retrieve such information given proper access permissions.
2. *Activation of communication channel(s).* This involves creating a standard for setting up communication channels. For example, activating a particular communication channel may result in launching the application that can handle the particular channel. Hence, activation of an H.323-compliant communication channel may result in a launch of a communication application such as an H.323-compliant phone. Having been launched, the phone application starts to listen for incoming connections or connects to a specified endpoint as needed.
3. *Communication via the channel(s).* This involves creating standards for the communication channels themselves. Some already exist, such as e-mail for non real-time communication, or H.323 for real-time audio, video, and data sharing. Others have yet to be defined such as instant messaging and shared virtual spaces. Furthermore, as mentioned previously, for some communication channels there may be little or no reason to create multiple interoperable clients. Most multi-player games fall into this category. Those communication channels may not benefit from standardization and need not be standardized.

A number of related standards efforts and protocols are worth mentioning here.

1. *Instant Messaging and Presence Protocol (IMPP) working group.* The IMPP IETF working group's charter includes standardization of presence detection and instant messaging protocol(s). Being able to determine the availability of endpoints or users and the IP addresses currently assigned to them will help in determining the availability of communication channels for a particular person, group of people, or endpoint(s). However, being able to enumerate the personal communication channels available for a particular person, group of people, or endpoint(s) is not within the current scope of the IMPP working group. It will have to be addressed separately.
2. *Service Location Protocol (SLP) [5].* The Service Location Protocol allows for the publishing of services. It could be used to publish available communication channels. The first version was standardized in July 1997 and work is currently proceeding on a second version. It also has an existing implementation. However, it is not meant to be Internet scale, and it only deals with finding a service on an Intranet. Service activation is not handled by the protocol.

3. *H.323 standard for real-time multimedia communication*. *H.323* defines a standard for initiation and setup of real-time multimedia communication channels, as well as content transfer between two or more participants over the Internet. Within the *H.323* standards framework, a number of standards for the real-time exchange of audio, video, and data have been defined.

   For example, *H.323* addresses call setup (Q.931), call and media control (H.245), real-time media transmission (RTP), network statistics gathering (RTCP), audio codecs (G.711, G.723, G.729), video codecs (H.261, H.263), and data sharing (T.120). Furthermore, T.120 defines an extensible framework that applications can use to define and standardize their own real-time data-sharing capabilities.

   H.323 was approved by the International Telecommunications Union (ITU) in 1996. It has broad support from the industry, including support from Microsoft, Intel, and (formerly) Netscape.

4. *IDentity Infrastructure Protocol (IDIP)*. The IDentity Infrastructure Protocol, as described in a current IETF draft and white paper by Shingo Fujimoto and Dave Marvit from Fujitsu Laboratories (http://idi.fla.fujitsu.com), advocates IDentity Objects that represent users on the Internet whether they are logged on to the Internet or not. The protocol describes the communication between IDentity Objects to control the initiation of applications and services between users.

   The protocol is in an early draft stage and has not gained much support at this point in time, but the ideas behind it are applicable to the enumeration and activation of personal communication channels or services.

CONCLUSION

Personal communication over the Internet is an emerging field. Today, various real-time and non real-time communication channels exist. Much work has gone into providing standards for real-time communication over the Internet, including the standardization of real-time audio, video, and data sharing through the *H.323* standard. Although non real-time personal communication is dominated by e-mail and newsgroups, it has recently seen a flurry of activity within the area of virtual shared spaces.

Buddy lists, one of the most popular communication applications on the Internet, do not interoperate. They use proprietary protocols for their communication channels and to detect when users become available for communication.

Work is underway at the Instant Messaging and Presence Protocol IETF working group to standardize the way transiently connected users and their IP addresses can be found. However, this is not enough. There is also a need to standardize the discovery of personal communication channels or services, proprietary or not, as well as a need for standardizing their activation.

History has shown that flexible standards promote interoperability and innovation and are key to realizing the potential of Internet communications. Hence, there is a need to provide a standards-based infrastructure that new communication channels or services can plug into, allowing easier deployment and rapid acceptance amongst users. This can be done by providing standards for publishing, finding, and activating personal communication channels or services for users or endpoints.

In establishing new standards, there is a need to build on the work that has already been done in other related standards efforts. In particular, there is a need to leverage the *H.323* standard, as well as existing standards for real-time and non real-time communication channels such as e-mail, newsgroups, and IRC-based text chat. Furthermore, there is a continuing need to standardize communication channels for which there are currently no standards, including instant messaging and virtual shared spaces.

In summary, personal communication over the Internet can greatly benefit from Internet-scale standards for publishing, finding, and activating personal communication channels or services. Such standards should co-exist with and build on existing standards as much as possible. Furthermore, these standards should be lightweight and flexible enough to embrace all personal communication channels, including real-time and non real-time, as well as standard and proprietary channels.

ACKNOWLEDGMENTS

I thank Jim Edwards, Brad Needham, and Aaron Kunze for helping me understand the nature of Internet-based personal communication as it exists today.

REFERENCES

[1] Postel, J., "Simple Mail Transfer Protocol," RFC-821, USC/Information Sciences Institute, August, 1982.

[2] Horton, M., "Standard for Interchange of USENET Messages," RFC-850, USENET Project, June, 1983.

[3] Kantor, B. and Lapsley, P., “Network News Transfer Protocol,” RFC-977, 1986.

[4] Goland, Y. et. al., “HTTP Extensions for Distributed Authoring – WebDAV,” RFC-2518, February, 1999.

[5] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. Service Location Protocol, RFC 2165, July, 1997.

[6] Oikarinen, J., Reed, D., "Internet Relay Chat Protocol," RFC-1459, May, 1993.

AUTHOR'S BIOGRAPHY

Christian Dreke received his M.S. degree in computer science from the University of Virginia in 1996. He joined Intel as a software engineer also in 1996 and has been working in the Communication Architecture Lab for three years. Most of his work at Intel has involved developing software to simplify personal communications over the Internet.

His e-mail address is Christian.Dreke@intel.com.

Supplementary Services in the H.323 IP Multimedia Telephony Network

Vineet Kumar, Intel Architecture Labs, Intel Corporation

Index words: H.323, H.450, packet telephony, IP telephony, multimedia conference, video conference, audio conference, supplementary services

ABSTRACT*

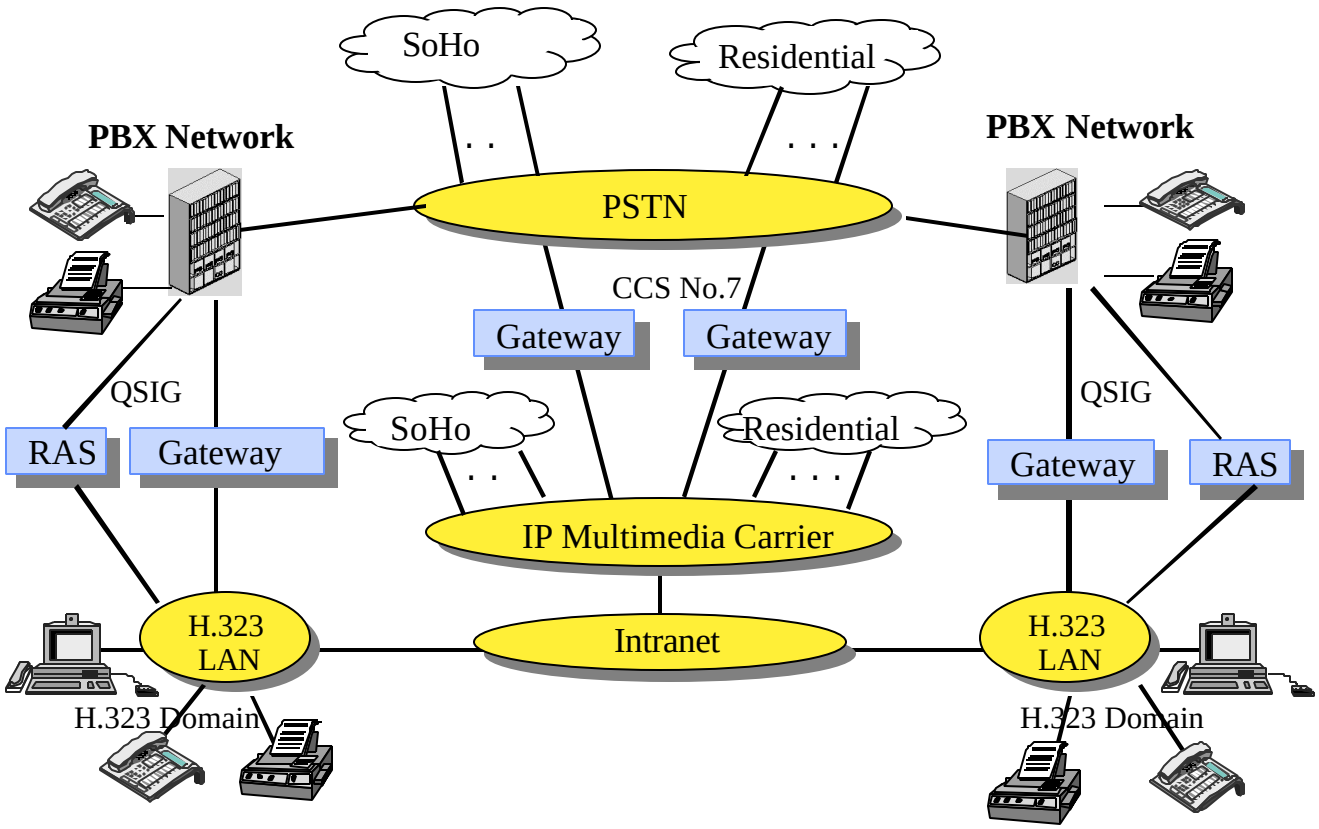
With the emergence of multimedia on packet networks, traditional telephone services are being developed for H.323 IP multimedia telephony networks. However, the architecture for signaling is different from the traditional centralized switch model in voice telephony. This paper describes the architecture and design of supplementary services in H.323 and compares the architecture to that of the switch model. The architectural model is peer-to-peer, the protocol design is based on the QSIG standards, and the services can be designed using a multi-tier approach. The peer-to-peer model reduces the dependence on the network to routing the signals and data; this is in sharp contrast to the traditional telephony switch model. QSIG is available as a worldwide standard (ISO/IEC JTC1) for private integrated services digital network (ISDN) telecommunication networks. Using the QSIG protocol design reduces the complexity involved in inter-working with the traditional circuit-switched equipment that uses QSIG. The multi-tier approach of designing services allows for complex services to be easily developed using building blocks consisting of simple services.

INTRODUCTION

The ITU-T H.323 [1, 2] series of recommendations describes terminals, equipment, and services for multimedia communication over packet-based networks (e.g., IP networks), and it covers the protocols necessary for their operation and for inter-connection with circuit-switched networks. H.323 terminals and equipment may carry real-time voice, data, and video, in any combination.

An example of the H.323 multimedia network is shown in Figure 1 where the H.323 network is interconnected to a legacy circuit-switched network.

Figure 1 shows some aspects of both the carrier and the corporate network along with the legacy circuit-switched network. Native H.323 calls for voice, facsimile, and other multimedia can be made within the H.323 domain; between H.323 domains using the intranet; within the H.323 domain in a carrier network; between subscribers of the H.323 domain in a carrier network and a corporate network; and between virtual private network (VPN) subscribers of the H.323 domain in a carrier network (Virtual Centrex Service). Inter-network H.323 calls can be made between PBX extensions and corporate H.323 users via the QSIG gateway; between PBX extensions belonging to different PBX networks via the H.323 networks for the purpose of toll by-pass; between residential public switched telephone network (PSTN) subscribers and corporate H.323 users via the common channel signaling system No. 7 (CCS No. 7) gateway; between PSTN and multimedia services network via the CCS No. 7 gateway; and between two PSTN subscribers via the multimedia services network using CCS No. 7 gateways for the purpose of toll by-pass.



Legend: SoHo = Small-office / Home-office
RAS = Remote Access Service
CCS No.7 = Common Channel Signaling System No. 7
QSIG = D-Channel signaling protocol at Q reference point for PBX networking

Figure 1: Example of an H.323 network interconnected to the legacy circuit-switched network

The supplementary services are covered in the H.450 [3] series of recommendations. In this paper, the architecture of supplementary services is described. Since the architecture is sharply different from that of circuit-switched networks, comparisons are made whenever appropriate.

ARCHITECTURE

The key elements of H.450 supplementary services are protocol based on ISO QSIG, peer-to-peer signaling, and a multi-tier approach to developing services.

H.450 Based on ISO QSIG

As services in the multimedia networks are deployed, they will have to inter-work with the services in the circuit-switched installed base. Being based on QSIG enables simple inter-working with the widely deployed installed base of private telecommunication networks that are QSIG based. Also, being based on QSIG enables smooth migration from the current installed PBX networks to H.323 multimedia networks.

H.450 Based on Peer-to-Peer Signaling

The H.450-based peer-to-peer signaling is different from the third party signaling of the switch. This difference results in changes to the way services are deployed, provisioned, charged, and to how incompatibility is resolved.

Service Deployment

The H.450 model is like the Internet model where intelligence resides at the ends/edges and the network simply routes the packets. The end device can be a PC or any IP phone, and the edge device is a PBX or a consumer gateway that resides at home. The state of the calls (ringing, busy, waiting, held, etc.) is also distributed to the ends/edges. This is in sharp contrast to the traditional switch model where intelligence and state of all calls is in

the network, and the ends/edges are simple phones running a stimulus-response protocol.

In H.450, services are deployed in much the same way as any other software package that is purchased from a store or downloaded from a Web site and installed on the end/edge device. Services can be developed by any vendor and sold directly to the end-user for deployment. In this environment, it is most likely that a huge industry will develop that will result in tremendous innovation in creating new services. In the switch model, services are deployed at the switch, which may result in other parts of the network being upgraded by the service provider before it is made available to the user. This involves a huge up-front cost for the service provider before any return on investment can be achieved from the users of the service. However, deploying services in a centralized location reduces some complexity as compared with the distributed nature of deployment in H.450.

Service Provisioning

In H.450, services are provisioned automatically by running the software package containing the services. In the switch model, services are provisioned by the service provider at the switches and obtained by the users through the use of DTMF and ringing tones.

Service Charging

Due to differences in the deployment of services, the model for charging is also different. In the H.450 model, the user pays for unlimited use of the services software up front. In the switch model, the service provider usually charges the user on a monthly basis. In H.450, the service provider has the option of imposing a small charge on the user for using the network for signaling, if the signaling is routed and monitored via the service provider's gatekeeper.

Service Incompatibility

In H.450, incompatibility of services between clients is resolved through the use of capabilities. The clients exchange their capabilities, and the services that are within the capabilities of both the clients can be executed. In the switch model, the more capable switch executes services on behalf of the less capable switch.

Multi-Tier Approach to Develop Powerful Services

One of the strengths of H.450 is that it enables a multi-tier approach to the development of services. Basic services consist of building blocks or primitives from which more complex services are developed. Compound services are developed from two or more basic services. Both basic and compound services are used by applications to provide multimedia services to the user. Examples of basic services supported in H.323 include multiple call handling, call transfer, call forwarding, call park and pickup, call waiting, message waiting indication, and n-way conference. Examples of compound services include consultation transfer and conference out of consultation. Consultation transfer uses call hold, multiple calls, and call transfer. Conference out of consultation uses call hold, multiple calls, and n-way conference.

BASIC SERVICES

The basic services that have been standardized in H.323/H.450 [3] are briefly described below.

Multiple Call Handling

This service [1] allows a multimedia client to handle multiple calls simultaneously.

Call Transfer

This service [4] enables the served user A to transform an existing call (user A to user B) into a new call between user B and user C selected by user A. User A may or may not have a call established with user C prior to Call Transfer.

Call Forwarding

This service [5] is also known as Call Diversion and it comprises these services: Call Forwarding Unconditional, Call Forwarding Busy, Call Forwarding No Reply, and Call Deflection. It is applied during call establishment, and it diverts an incoming call to another destination alias (e.g., telephone number, IP address, e-mail address) address.

Call Forwarding Unconditional permits a served user to have incoming calls, which are addressed to the served user's number, redirected to another number.

Call Forwarding Busy enables a served user to have calls, which are addressed to the served user's number and meet busy, redirected to another number.

Call Forwarding No Reply enables a served user to have calls, which are addressed to the served user's number and for which the connection is not established within a defined period of time, redirected to another number.

Call Deflection permits a served user to respond to an incoming call offered by the served client by requesting diversion of that call to another number specified in the response. This request is only allowed before the called user has answered the call.

Applications can be developed to allow the above variants of call forwarding to operate on all calls, or only on selective calls that fulfill specific programmed

conditions or conditions manually entered by the user. For example, forwarding can be made dependent on various conditions such as the state of the called party, busy, no-reply, absent; the caller identification; the time of the day; the day of the week; etc. For each scenario, the user can program the forwarding of incoming calls to different destination addresses. For example, a user can program his/her client to forward all incoming calls between 8 a.m. and 5 p.m. on weekdays to his/her office phone, calls on weekends from specific Caller IDs to ring on his/her home phone, and all other calls to be forwarded to his/her voice mail. Programming of the destination address can be done locally at the home client or by remote programming via a connection to the home client. Such a rich service is not easily available in traditional telephony.

Call Hold

This service [6] enables the served (Holding) user A to put user B (with whom user A has an active call) into a hold condition (Held User) and subsequently to retrieve that user B again. During this hold condition, user B may be provided with music and/or video on hold. The served (Holding) user A may perform other actions while user B is being held, e.g., consult with another user C.

Call Park and Pickup

Call Park [7] is a service that enables the parking user A to place an existing call with user B (Parked User) to a parking position. The call can later be picked up by retrieving the parked party from the same terminal where the park took place or from another terminal. Call Pickup is a service that enables the picking-up user to pick up a parked call. Upon successful invocation of Pickup, the picking-up user is connected with the parked user B.

Services such as Call Park/Pickup are used in the Automatic Call Distribution environment where calls are not directed to a specific user/client but to, for example, the first available agent with some specific skill. Such services are best implemented in the Feature Server, which then interfaces with a group of clients.

Call Waiting

This service [8] permits a served user who is busy with one or more calls to be informed of an incoming call with an indication. The user then has the choice of accepting, rejecting, or ignoring the waiting call. The user calling the busy party is informed about the call waiting condition.

Message Waiting Indication

This service [9] provides a general-purpose mechanism by which a user can be advised that messages intended for that user are available. A variety of message types are supported, such as voice mail, fax, and telex. In one of its simplest forms, when a message is left for a user, a Message Center sends a notification to the Served User, where a Message Waiting LED is lit. Additional information provided by the notification mechanism allows the Served User to know the number of messages that are waiting, the types of messages, the subjects of the messages, and the priority of the highest priority message.

N-Way Conference

This service [1] allows a multi-party conference to be established. This can happen as a result of two or more simultaneous calls merged into one conference or as a result of an initial two-party call later expanded to a conference. The limit on the number of participants in a conference is usually based on the policy of the entity hosting the N-Way Conference.

COMPOUND SERVICES

Compound services, on the other hand, use basic services to provide more powerful services to the end user. Two such services, Consultation Transfer and Conference Out of Consultation, are described below in detail.

Consultation Transfer

In Consultation Transfer, the user can perform three operations: (1) put a multimedia call on hold and retrieve it later, (2) call another person and optionally alternate between the two calls, or (3) transfer the call. The operations involve three supplementary services: Call Hold, Multiple Call Handling, and Call Transfer.

Call Hold. During consultation, the first call between A and B needs to be held so that it does not interfere with the second conversation between A and C. This is shown in Figure 2 In this case, the holding client A stops sending multimedia information (i.e., voice, video, and data) to the held client B so that the held party B cannot hear or see what happens during the consultation. At the same time, the holding party A is not able to see or hear the held party B. The simplest form of hold is called *near-end Hold*. In this case, the holding client A stops sending media, causing silence and “frozen picture” on the held client B. The other form of hold is *remote-end Hold*. In this case, the holding client A sends multimedia information, such as commercials, to the held client B (often called “music/video/document on hold”). At the same time, the holding client A may place the first call on mute and stop receiving multimedia packets from the held client B. The holding client A informs the held client B by sending a notification message via the signaling path.

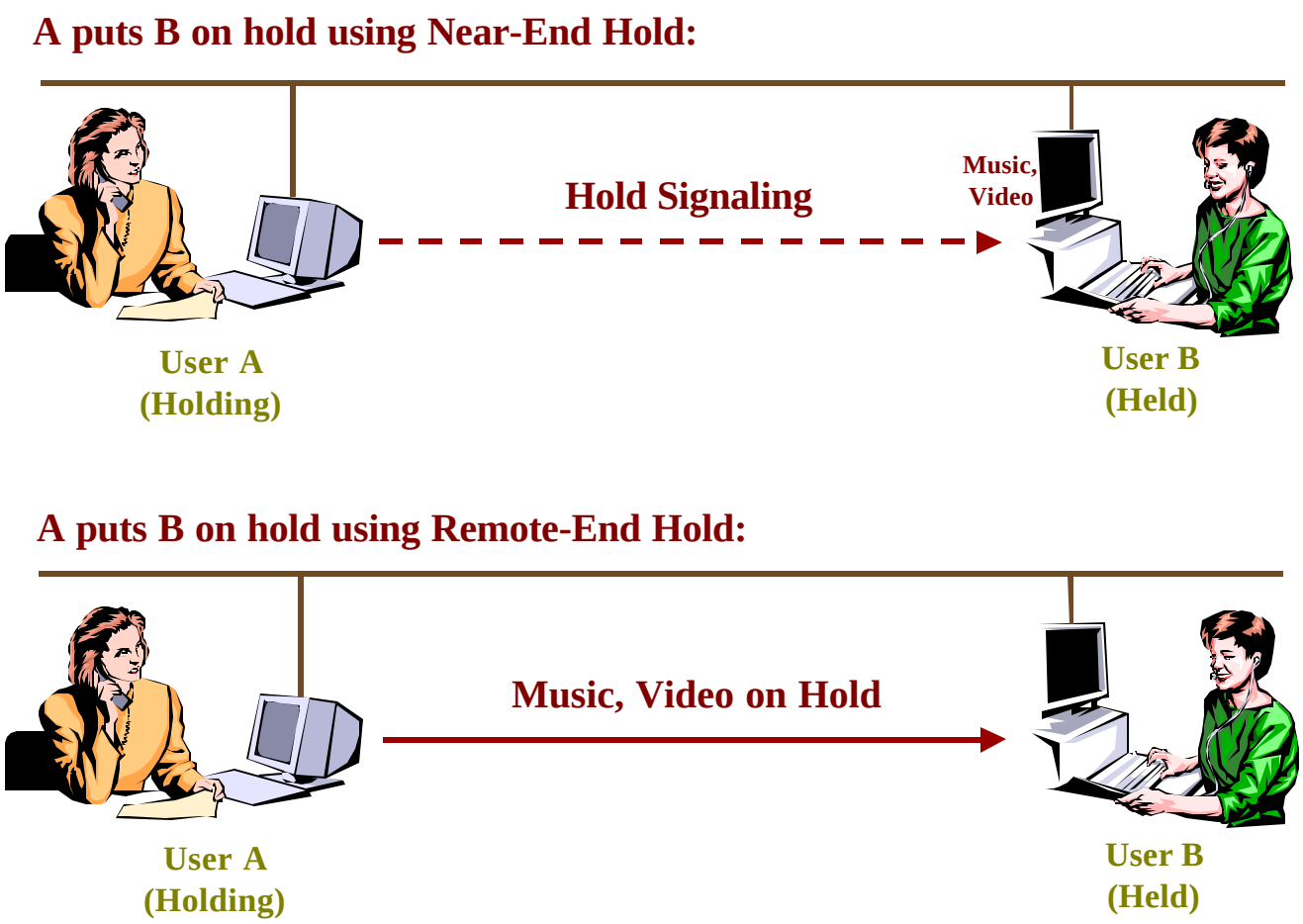


Figure 2: Call hold supplementary service

Multiple Call Handling. While the first call between A and B is held, the initiating client A establishes a second call to client C of the consulted party as shown in Figure 3. Since there is no hard limit on the number of H.323 calls a client can make simultaneously, this second call can be established in the same manner as the first call; in other words, it can be established as a basic call. Since party A is in two calls, one being held and another being active, he/she may swap between these two calls by putting the active call on hold and retrieving a previously held call.

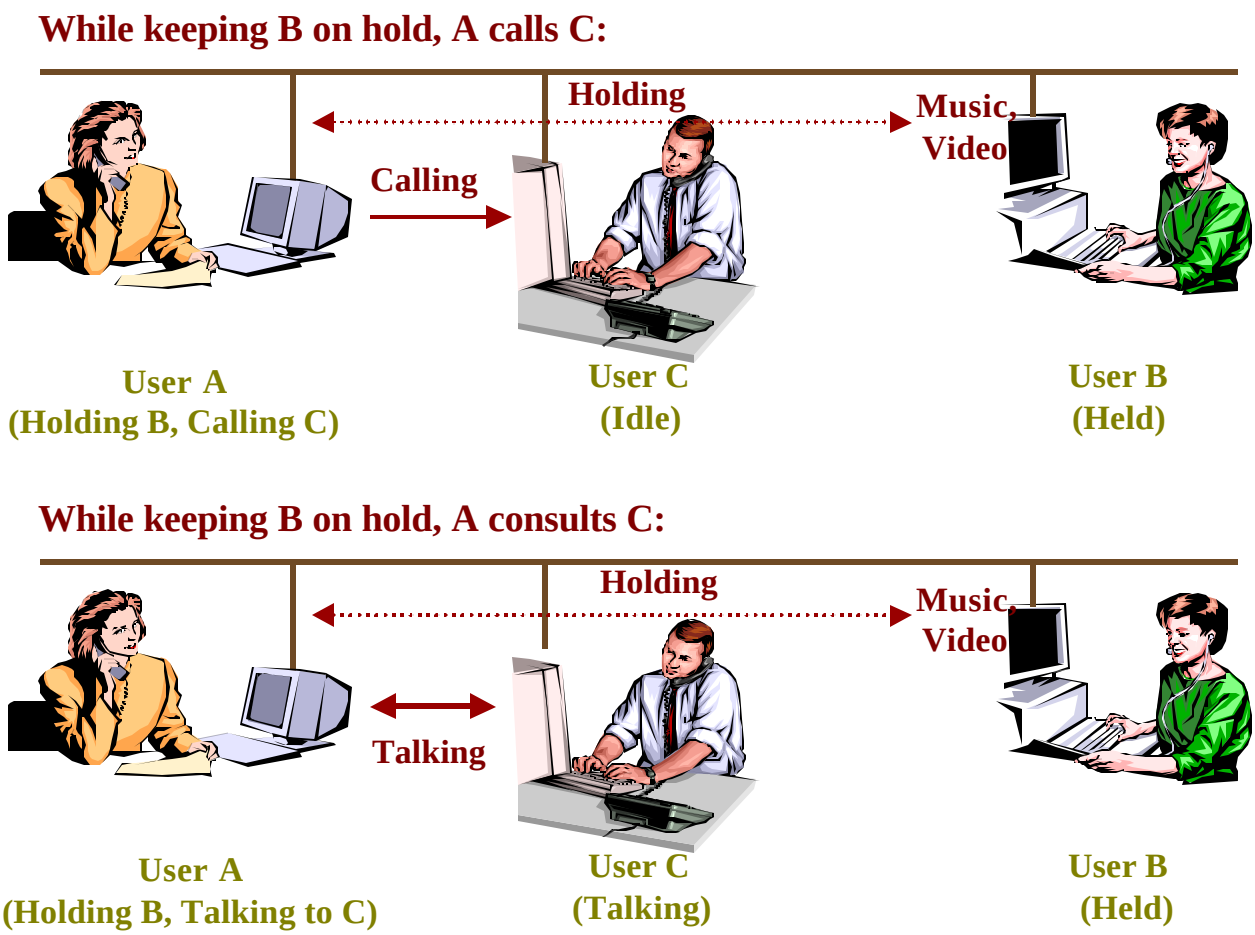


Figure 3: Multiple call handling supplementary service

Call Transfer. Call transfer is a basic service that enables party A to transfer an existing call with party C from his/her client A to client B as shown in Figure 4. Usually a multimedia conversation precedes the transfer, but it is not necessary. This transfer is performed in a single-step, i.e., the transferring client A sends a signaling message to the transferred client C requesting a new connection to be established directly from client C to the transferred-to client B. The transferred client C then establishes an independent connection to the transferred-to Client B. The first call between clients A and B is dropped if the transfer is successful.

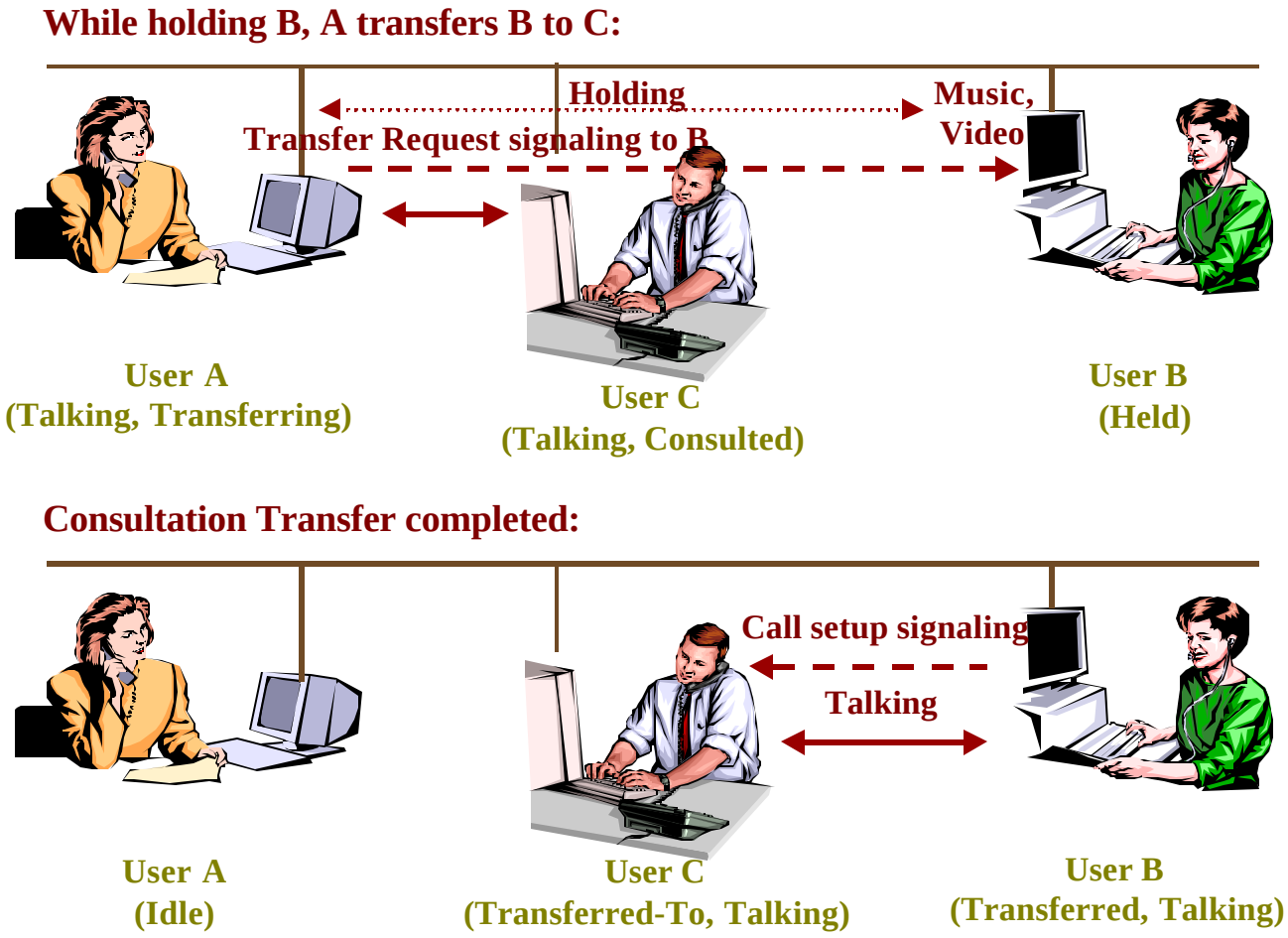


Figure 4:Call transfer supplementary service

CONFERENCE OUT OF CONSULTATION

In Conference Out of Consultation, the user can perform three operations: (1) put a multimedia call on hold and retrieve it later, (2) call another person and optionally alternate between the two calls, or (3) merge the calls in one conference call. The operations involve three supplementary services: Call Hold, Multiple Call Handling, and N-Way Conference. The services involving Hold, and Multiple Calls were discussed in the Consultation Transfer scenario where the call between client A and B was on hold and the call between A and C was active. The third service is described below.

N-Way Conference. This is a basic service that enables party A to merge the two existing calls between A and B and A and C into one conference call between A, B, and C. The signaling consists of client A transferring the calls between B and C to the Conference Server, and then making his/her own call to the Conference Server as shown in Figure 5. Previous calls between A and B and A and C are dropped if the conference is successfully established. In H.323, the address of the Conference Server is declared by each client as part of its capability during call setup time. The Conference Server could be resident in the client or be a resource on the network.

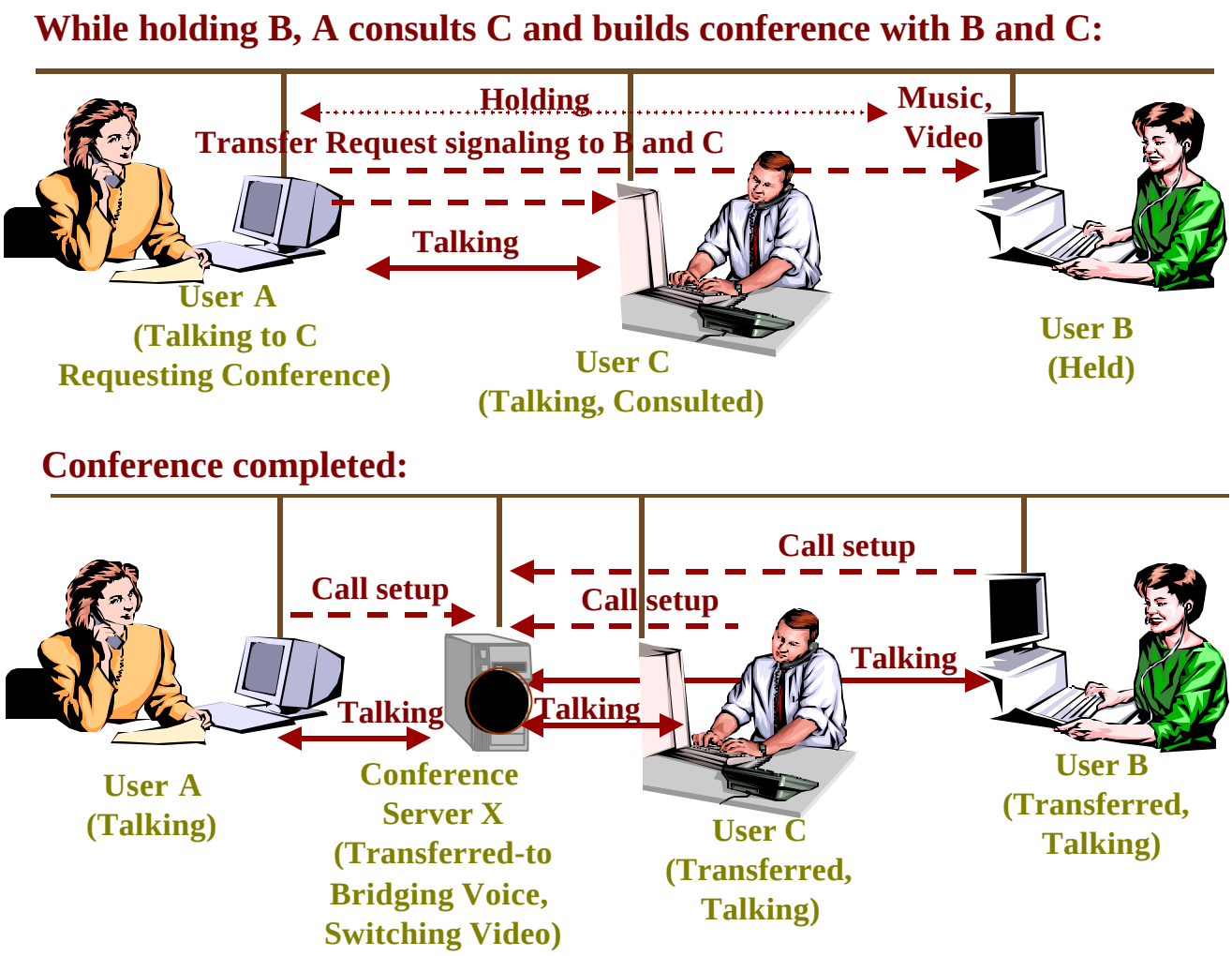


Figure 5: N-way conference supplementary service

CONCLUSION

International standards have been developed for deploying services on packet networks that rival services offered through circuit-switched networks. The architecture is suitable for use by both corporate networks and new-generation telcos. The deployment of H.323 has already begun in corporate networks. Savings in operational and upgrade costs are compelling reasons for unifying the voice and data networks. Unified networks provide richer services such as unified messaging involving multimedia instead of having voice-mail in the voice networks and e-mail in the data networks.

Even though the H.450 basic services developed so far mimic those already available in the circuit-switched corporate environment, the power of H.450 is in the multi-tier approach of developing services and in the distribution of service logic. More powerful and innovative services can be developed simply by using the basic services as building blocks. Since the service logic is distributed in the desktop phones as compared to being centralized at the PBX for the circuit-switched networks, a software PBX can be embedded in each phone. This removes the single point of failure at the centralized PBX and provides high scalability and fault tolerance in the H.323 network.

Since H.323 networks will coexist with the circuit-switched networks for a long period of time, using QSIG provides for smooth migration and simpler gateways to inter-work between the two networks.

REFERENCES

[1] "Packet-Based Multimedia Communications Systems," ITU-T Recommendation H.323, Geneva, Switzerland, January 1998.

[2] "Call Signaling Protocols and Media Stream Packetization for Packet-Based Multimedia Communications Systems," ITU-T Recommendation H.225.0, Geneva, Switzerland, January 1998.

[3] "Generic Functional Protocol for the Support of Supplementary Services in H.323," ITU-T Recommendation H.450.1, Geneva, Switzerland, January 1998.

[4] "Call Transfer Supplementary Service in H.323," ITU-T Recommendation H.450.2, Geneva, Switzerland, January 1998.

[5] "Call Diversion Supplementary Service in H.323," ITU-T Recommendation H.450.3, Geneva, Switzerland, January 1998.

[6] "Call Hold Supplementary Service in H.323," ITU-T Recommendation H.450.4, Geneva, Switzerland, September 1998.

[7] "Call Park and Call Pickup Supplementary Services in H.323," ITU-T Recommendation H.450.5, Geneva, Switzerland, September 1998.

[8] "Call Waiting Supplementary Service in H.323," ITU-T Recommendation H.450.6, Geneva, Switzerland, September 1998.

[9] "Message Waiting Indication Supplementary Service in H.323," ITU-T Recommendation H.450.7, Geneva, Switzerland, September 1998.

AUTHOR'S BIOGRAPHY

Vineet Kumar is a Distinguished Member of the technical staff at Intel Architecture Labs. His current interests are in the standardization, implementation, and commercialization of IP multimedia telephony. He has been active in the standardization of H.323 and has served as editor of several H.323-related recommendations at the ITU-T. He received his B.S. degree in 1980, his M.S. degree in 1982, and his Ph.D. degree in 1985, all from Iowa State University, Ames, Iowa. His e-mail address is vineet.kumar@intel.com.

Reactive Mechanisms for Recovering Audio Performance in Multimedia Conferencing Over Packet Switched Networks

Hani ElGebaly, Intel Architecture Labs, Intel Corporation

Index words: RTCP, H.323, congestion, multimedia, jitter, packet loss

ABSTRACT

The impact of multimedia traffic on the performance of networking applications is significant because the vast majority of currently available tools send data at a rate that does not depend on the state of the network. This is unlike rate-controlled applications, such as Transport Control Protocol-based (TCP) applications, which adjust their output rate and bandwidth requirements according to the state of the network. Audio is the most important component in a multimedia session. It is thus important to grant audio the maximum attention even if it necessitates sacrificing the throughput (but not the quality of service) of other less important applications. An example of such a sacrifice is to reduce the video bit-rate if the audio quality drops. Reducing the bit-rate may or may not be observed by the user. A video source may adapt to this change by changing the target frame rate if the drop in bit-rate is significant.

In this paper we address feedback algorithms for dealing with audio loss problems. We propose a modification to a Real-time Control Protocol-based (RTCP) congestion control algorithm that is sensitive to audio performance. We analyze the RTCP-based congestion control technique and present its limitations. We then uncover a relationship between audio jitter and audio packet loss. Finally, we present a novel scheme for predicting loss using jitter information.

INTRODUCTION

Advances in computer and communication technology have stimulated the integration of digital audio and video with computing. This integration led to the development of multimedia applications such as video conferencing and multimedia collaboration. Many of these applications are targeted to run over packet-switched networks such as the Internet. Packet-switched networks with non-guaranteed quality of service do not seem suitable for real-time traffic such as multimedia conferencing.

Internet switches do not treat network traffic in a fair way. Packets are routed independently across shared routers and switches. These switches do not pay attention to the type of packet, packet loss, or latency sensitivity. As parts of the Internet become heavily loaded, congestion can occur. Congestion may led to buffer overflow and packet loss. It may also lead to packet delay as packets take longer to process. Latency may seem acceptable for some applications such as e-mail and file transfer. For real-time applications, data becomes obsolete if they do not arrive in time. In addition, real-time applications can be quite sensitive to loss. Furthermore, since packets are routed independently across shared routers and switches, transit times may vary significantly. Variation in transit delay is called jitter, and jitter is very disturbing to real-time applications, especially to audio playback. It significantly reduces speech intelligibility to the ear and causes choppiness and breakup in the stream. Consequently, real-time applications deliver poor quality during periods of congestion over shared bandwidth networks such as the Internet.

Several researchers have addressed these Internet problems and have proposed infrastructure enhancements or modifications to the current Internet TCP/IP-based infrastructure.

Infrastructure enhancements call for new technologies, for example, fiber optics [2], better and faster switch architectures [1], and new protocol layers such as B-ISDN [7] ATM [12], etc.

Infrastructure modifications call for implementing congestion control mechanisms in the network routers that reward rate-controlled applications and punish aggressive ones. These modifications require that router software and the current infrastructure of the Internet be modified. The purpose of these modifications is to

provide ways of enforcing resource reservation, quality of service bounds, and recognition of real-time traffic. For example, the RSVP [14] protocol maintains a specific quality of service for participants in a communication session. In this model all routers along the data path keep a record of all current service requests and check all packets to see if they need special treatment. Another example is the Internet Protocol (IP) type of service (TOS) byte that was intended to express eight precedence levels of service (in an increasing precedence order). Unfortunately, this service is underutilized by implementers. The IETF redefined those TOS bits in the IP protocol as a differentiated services field [3]. The differentiated services field will contain a six-bit flag assigned by the server at the edge of the network or by a classifier built into a router to grant the packet a class of service based on its priority.

Other solutions for the congestion problem include implementing TCP-friendly rate control algorithms, jitter compensation at the receiver, and forward error correction fault-tolerant algorithms.

Undoubtedly new technologies will evolve over the years. The deployment of these technologies may take some time, but eventually will make it to the implementation phase. Indeed many of these technologies have already been running for years over private networks exhibiting outstanding performance. Yet, solutions for current networks are essential since they will persist for a while as a vital communication media for many end users.

In this paper, we discuss two TCP-friendly rate control algorithms that save the audio quality in a multimedia connection. Both algorithms are based on reactive mechanisms used to recover media flow performance degradation caused by the effects of shared bandwidth traffic. We present an overview of feedback mechanisms based on the Real-time Control Protocol (RTCP). We uncover the limitations of RTCP feedback on applications connected to the Internet through narrow bandwidth pipes. We propose an alternative approach to the RTCP feedback algorithm that predicts and prevents the loss of audio packets before it occurs, based on local computation of incoming audio jitter. These approaches improve the audio performance significantly in multimedia conferencing sessions.

LOCAL AND REMOTE SCHEMES FOR CONGESTION HANDLING

The poor performance of multimedia conferencing over the Internet can be attributed to two main factors: local- and remote-induced effects. Local effects are induced by bandwidth sharing between different media components, operating system limitations, or poor design. Remote effects include all Internet-related problems such as unfairness, non-guaranteed quality of service, congestion, etc.

Consequently, solutions to overall audio latency and jitter in a conferencing session are either local to the host (local schemes) or based on feedback from the remote conferencing peer (remote schemes), or a combination of both.

Local schemes address issues such as careful scheduling of video and audio traffic at the host before media components reach the shared bandwidth network layer. The objective of these schemes is to prevent video from oversubscribing to the available bandwidth, thereby degrading the performance of the audio stream.

Remote schemes rely on statistics such as loss, jitter, latency, timestamps, etc. exchanged between network terminal peers. A terminal, upon receiving these statistics, will decide to adjust the cumulative throughput of one or more media channels in order to make up for loss, latency, or jitter symptoms. This class of feedback scheme is called Remote Feedback Local Control (RFLC). An example of this kind of feedback is the RTCP reports used in H.323 [9] conferencing and general multicast streaming. Alternatively, terminals can compute these statistics based on the actual stream of media data and provide Flow Control commands to remote terminals to limit throughput of the degrading media type. In this paper, we discuss this class of feedback scheme known as Local Feedback Remote Control (LFRC). Our objective is to devise a scheme that can predict the occurrence of audio loss and take appropriate actions to prevent it.

MAIN CAUSES OF POOR AUDIO PERFORMANCE

There are mainly three reasons for poor audio performance: packet loss, delay, and jitter. Packets may be lost in the network and never arrive at the receiver. Audio loss is manifested as lost phonemes in a word or even lost words in a sentence. It can also be manifested as breakage in the continuity of audio playback. Packet latency leads to audio frames arriving late. Latency disturbs speech continuity and interactivity. Packet inter-arrival jitter causes choppiness of audio playback at the receiver.

Packets get lost due to buffer overflow or due to bit errors. The probability of bit errors is very low on most packet-switched wired networks. Hence, loss is mostly induced by buffer overflow [11]. Buffer overflow can happen on a congested link (Internet intermediate switch) or at the network interface of the workstation. The network

interface loss on low speed links can be due to a narrow bandwidth pipe supplied by the Internet service provider. Controlling the bit-rate of the media stream can alleviate the loss problem.

Packets are delayed and jittered because of long waiting time in Internet intermediate switch queues, long routes, and heterogeneous link speeds. Unless Internet infrastructure substitutes the current best effort routing algorithm with another fair and multimedia-aware algorithm, there is little hope for conquering the delay problem effectively in today's Internet.

Packet Loss and Latency Contributors

Modem Connection Latency

The latency contributed by a modem connection is composed of the per-byte transmission time (modem bandwidth) and a fixed software and hardware overhead.

The transmission time depends on the transmission rate of the modem. The fixed hardware and software overhead varies between different architectures and platforms. One delay factor, which is attributed to that overhead, is grouping of data. There is a modem wait time as the modem tries to group data into blocks and perform compression and automatic error correction. To get effective compression and error correction, modems must work on large blocks of data. This means that characters must be buffered until a sufficient block is built for the modem to work on efficiently. Data are not sent until processed by the modem's compression/error correction engine. This adds to the latency of data as they pass through the modem. In addition, the modem does not know the kind of data being sent, and consequently cannot use the best data-specific compression algorithms. For example, multimedia data are usually compressed before they pass through modems. Modem compression in this case is futile. In fact, compression may significantly affect the timeliness of the latency-sensitive data transferred through the modem.

For a typical modem link, the latency due to modem software and hardware overhead is usually about 100ms. The transmission time of 10 characters over a 33kbit/sec modem link time would theoretically be 80 bits / 33000 bits per second = 2.4ms. The actual time taken because of the compression overhead is 102.4ms. This is because of the 100ms latency introduced by the modems at each end of the link.

To enable small chunks transfer, there is a timeout value before the modem starts processing a data block. Using this timeout value, modems can avoid waiting indefinitely for the large block and so avoid causing huge delays to the peer. Hence, modem hardware and software overhead is a significant contributor to latency when connected to the Internet via modems through ISPs.

Internet Service Providers Bottleneck

The Internet service provider is the means by which the home user and many businesses hook up to the Internet. Users connect to their ISPs usually over Plain Old Telephone Service (POTS) lines via modems. An Internet provider in turn must connect to another wholesale Internet provider. The connection of an ISP to the wholesale provider is called the ISP pipe. Most ISPs anticipate relatively low bandwidth activity from their users such as Web browsing, reading information, etc. Multimedia conferencing is rarely taken into account. For example, Table 1 shows how an ISP pipe size may map to the number of ISP users [5]. Table 1 does not take into account if users are engaged in multimedia conferencing sessions. In the future, however, ISPs will have to satisfy the requirements of their conferencing clients.

Typically, each conferencing client logged on to an ISP via a 33.6kbps modem will chew up at least 8-10kbps for audio and 16-22kbps for video of the available bandwidth. Note that both numbers almost add up to the dialup connection bitrate available. This leads to a significant amount of latency and loss during the conferencing session. Further, most ISPs are oriented to downloading (down-streaming). Their down-streaming pipe is usually wider than their up-streaming pipe. Hence, packet loss is more significant when a conferencing client is sending media data upstream.

Pipeline Size	Number of Simultaneous Dial-up users
28.8K modem connection	3
56K DSO leased line	7
64K ISDN connection	8
128K ISDN connection	15
256K Fractional T1	40
512K Fractional T1	100+
Full T1	300+

Table 1: ISP pipeline size versus number of users

An overview of the streaming protocol that was used for this study is given in the next section.

H.323, RTP, AND IP TELEPHONY

IP telephony has become an important driver for packet-based communications. The H.323 recommendation developed by the ITU has provided a good basis for establishing a universal IP voice and multimedia communication in large, connected networks [8]. By using Q.931 as its basis for establishing a connection, H.323 allows for relatively easy bridging to the public switched telephone networks (PSTN) and circuit-based phones. The required voice codec of G.711 also allows for easy connections to the legacy networks of telephones. As the standard evolved, additions and extensions made it more suitable for IP telephony. H.323 has been declared the standard for Voice over IP communication by international standards' bodies such as the European Telecommunication Standard Institute (ETSI).

H.323, similar to a few other protocols, uses the Real-time Protocol (RTP) for data streaming. RTP is a real-time protocol for multiplexing media components in a media stream connection. It does not offer any form of reliability nor does it ensure real-time delivery. The protocol is real-time in the sense that it provides functionality suited for carrying real-time content, e.g., a timestamp and control mechanisms for synchronizing different streams with different timing properties.

RTP assigns timestamps for multimedia packets such that the timing order is restored at playback side. It also assigns sequence numbers in order to be able to detect losses. The RTP packet header size is 12 bytes. It carries information about payload type, source identifier, time stamp, and sequence number. Although RTP is an unreliable protocol, it provides hooks for adding reliability through fault tolerance, feedback reactive algorithms, and other reliability schemes as discussed in the following sections.

Along with RTP there is another protocol that is responsible for packaging information about connection quality statistics. This protocol is termed the Real Time Control Protocol and is discussed next.

REAL TIME CONTROL PROTOCOL – RTCP

RTCP is an associated protocol with RTP designed to handle the delivery monitoring service of the RTP protocol. It is based on periodic transmission of control packets, collected from all participants of a session, to all other participants. It provides feedback on the quality of media packets including timing information. All RTP senders generate a sender report. A sender report contains information useful for media synchronization as well as cumulative counters for packets and bytes sent. This information allows receivers to estimate the sender's data rate. Receivers generate receive reports for all video and audio sources they have heard from. These reports contain information on the highest sequence number received, the number of packets lost, a measure of the inter-arrival jitter, and the timestamps needed to compute the round-trip delay between sender and receiver issuing the report.

RTCP packets also contain a unique identification for their original sender via a source description (SDES) packet. This packet may contain user name, canonical name, e-mail, etc.

In the next section we propose an algorithm based on RTCP reports feedback that is a modification of Busse's algorithm [1].

RTCP FEEDBACK CONGESTION TECHNIQUE

Busse et al. [6] introduced a dynamic quality of service (QoS) control mechanism for multimedia applications based on RTCP feedback. Their technique is based on receiver reports delivered from receiving end applications to the source. This new algorithm is a modification of this approach. It is based on the use of an audio receiver report to adjust video bandwidth in a multimedia session between two peers. The objective is to provide audio with enough bandwidth to function with an acceptable QoS by sacrificing some of the video bandwidth.

The network is divided into distinct congestion states. Each congestion state is determined by the amount of audio packet loss experienced in the previous determined interval. The interval is the inter-arrival time between RTCP reports packets. The number of audio packets lost determines whether the network is in the unloaded, loaded, or congested state as depicted in Figure 1.

Assume that two H.323 terminals are engaged in a H.323 conferencing session. Both terminals stream audio and video to each other on separate UDP connections. Both terminals also exchange sender and receiver RTCP reports on separate UDP connections for both audio and video. The conferencing manager at each terminal, on receiving an audio RTCP receiver report (ARR) from a remote peer, does the following:

- Analyzes the statistics reported by the ARR for loss.
- Determines the network congestion state. A network can be in an unloaded, loaded, or congested state. The loss value obtained from the ARR determines the current network congestion state. This value is used to decide whether to increase, hold, or decrease the video bit-rate of the terminal.

- Adjusts video bit-rate according to the network state analysis. The user has the freedom to specify the minimum and maximum video bandwidth allowed. The user also has the freedom to specify audio parameters that determine the network's states and can adjust the video bit-rate accordingly.

Parameter values for determining the current state of the network are shown in Figure 1. λc is the lowest audio packet loss value beyond which the network becomes congested and video bit-rate has to decrease by δc. λu is the highest audio packet loss value below which the network is lightly loaded (or unloaded) and video bit-rate will increase by δu. During the loaded state, the video bit-rate is unchanged. The choice of the parameter values is subjective and may depend on the current condition of the network.

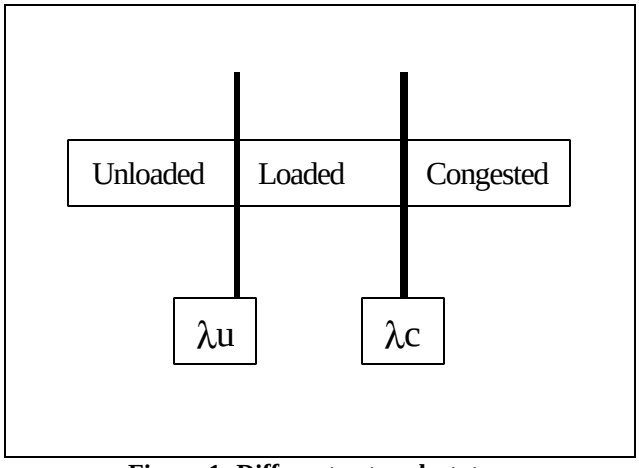


Figure 1: Different network states

We implemented the above algorithm on two Intel H.323-based Video Phone terminals. Both terminals were connected over the Internet via different ISPs. The maximum video bandwidth was set to 25 kb/s. Both terminals used G.723.1 as their preferred audio codec and H.263 as their preferred video codec. An audio interview file that lasted for approximately ten minutes was used as the audio input sample. We chose δc = 1 kb/s, δu = 0.5 kb/s, λc = 45 packets, and λu = 30 packets. The experiment was carried out during a known Internet rush hour to simulate the worst congestion scenario. Both terminals were running full-screen large format video (CIF) and low bit-rate audio.

Figure 2 shows the relationship between video bandwidth and audio packet loss as the session proceeds. It is observed that as audio loss increases beyond the congested state, video bit-rate decreases quickly, and consequently audio loss starts to improve. Video bit-rate starts to rise again whenever audio loss falls below the loaded state. The choice of parameters that determines the state of the network is user dependent. Our choice was based on statistics collected after running multiple sessions in a similar environment.

The algorithm increases the video bit-rate as long as the audio loss is below the network threshold and the network is in the unloaded state as depicted in Figure 1. Once the audio loss starts to increase, exceeding the network loaded state and entering the congested state, the algorithm reacts by decreasing the video bit-rate pulling the network back to the loaded state. Note that the rate of increase (slope of the curve) of the video bit-rate is slower than the decrease rate in order to gain more control over the audio congestion problem. The algorithm will maintain the video bit-rate at its current level as long as the loss rate is within the limits of the loaded network state.

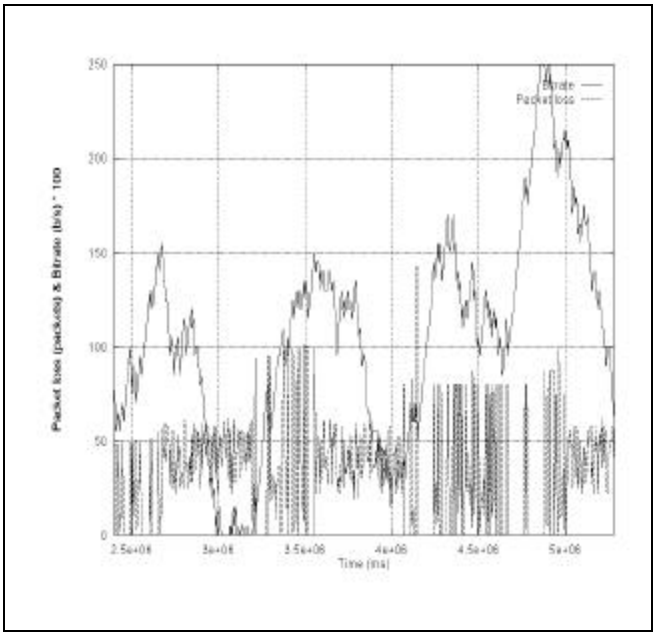


Figure 2: Video Bit-rate versus audio loss

There are several disadvantages to this approach. These disadvantages are discussed in the following section.

RTCP FEEDBACK LIMITATIONS

In order to keep RTCP overhead to a minimum, thereby better utilizing data packets, RTCP packet transmissions are limited. In general, no more than five percent of the total bandwidth of a media stream can be allocated to RTCP functionality [1]. On slow links with limited bandwidth, the inter-arrival time between successive RTCP receive reports may exceed seven seconds. This interval is quite long and may lead to slower response of adaptive schemes to network conditions. In other words, more audio loss than expected may occur before the terminal reacts by decreasing the video bit-rate. A more responsive approach is required to effectively control the

flow of all media streams. Hence a new feedback algorithm is required that does not wait for late statistics from the network and responds promptly to irregularities in the network.

Another limitation is the lack of criteria to determine accurate estimations of network state parameters. Internet home users may have relaxed requirements for the parameters; however, these parameters may not be acceptable to business users. The user has to rely on statistics for each particular case.

A third limitation to the RTCP feedback approach is that a possible QoS oscillation may occur where video bit-rate and audio loss follow a sinusoidal waveform. This phenomenon will make audio conversation unintelligible because of large peak losses. Video, on the other hand, will also suffer from poor performance because of reduced bit-rate. Long inter-arrival intervals between reports together with choice of control parameters contribute to this phenomenon.

A fourth drawback to the RTCP feedback approach is related to its periodic nature. Congestion or packet loss is a sporadic event; hence, it has to be treated by another non-periodic event. In fact, network congestion caused by audio loss, particularly in real-time applications, is short-lived and non-correlated [4]. Thus, reaction to loss should be prompt and instantaneous. In addition, waiting until loss happens and reacting afterwards may not be the best approach. A better approach is to provide a means to predict loss that informs the media terminal to slow down in order to avoid loss from happening. This approach is discussed in the following section.

RELATIONSHIP BETWEEN PACKET LOSS AND JITTER

An experiment was conducted using similar input and network conditions to those used in the case of the RTCP feedback experiment. We measured audio packet loss, inter-arrival time between packets, and inter-arrival jitter at the receiver. -The packet loss is measured as the difference between the sequence numbers of packets arriving at the receiver. -If the difference is greater than one, then loss occurred. —The inter-arrival jitter is measured as the difference in packet spacing at the receiver compared to the sender for a pair of packets as suggested in [1]. This is equivalent to the difference in the "relative transit time" for the two packets. The relative transit time is the difference between a packet's RTP timestamp and the receiver's clock at the time of arrival. This value is measured in milliseconds.

Figure 3 shows a plot of the inter-arrival time according to the receiver clock and audio packet loss, both against time. Figure 4 shows an extension in the time domain for the same plot. The audio packets used in this experiment carry four audio frames per packet. The frame length for the G.723.1 codec is 30ms. Hence, the packetization time is almost 120ms. Further, the audio session that was used for this experiment had very small (almost negligible) silence periods. Both Figures 3 and 4 show that change in the inter-arrival time of the audio packet was followed by a jump in packet loss. It is observed from the plots that the peaks of packet loss often occurred shortly after an abrupt change in the inter-arrival time of previous packets. Smaller changes in the packet inter-arrival times were followed by no or just small packet losses. The more significant the change in the inter-arrival time, the higher the loss value as shown in the figures. This was a motivation to pursue this observation further and find out if there is any relationship between audio inter-arrival jitter and packet loss.

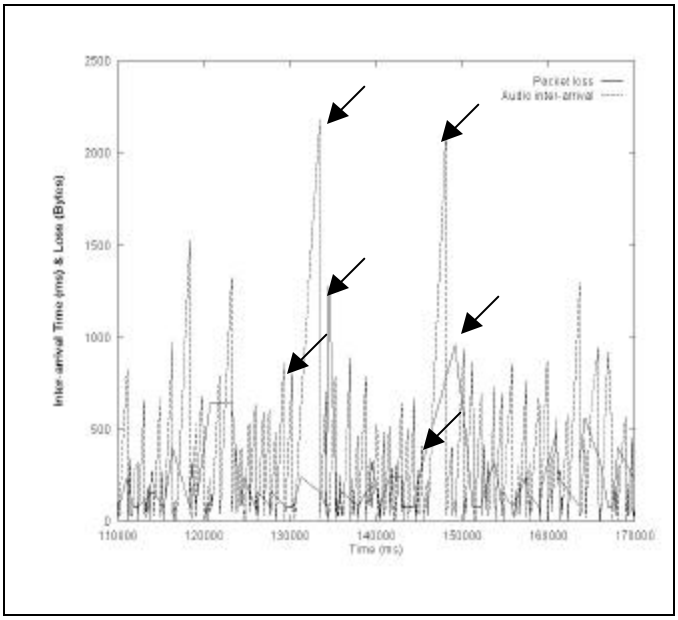


Figure 3: Audio receiver inter-arrival time and packet loss

We computed the audio inter-arrival jitter at the receiver. The audio jitter is computed as the rate of change of the inter-arrival time. The jitter value can be positive or negative.

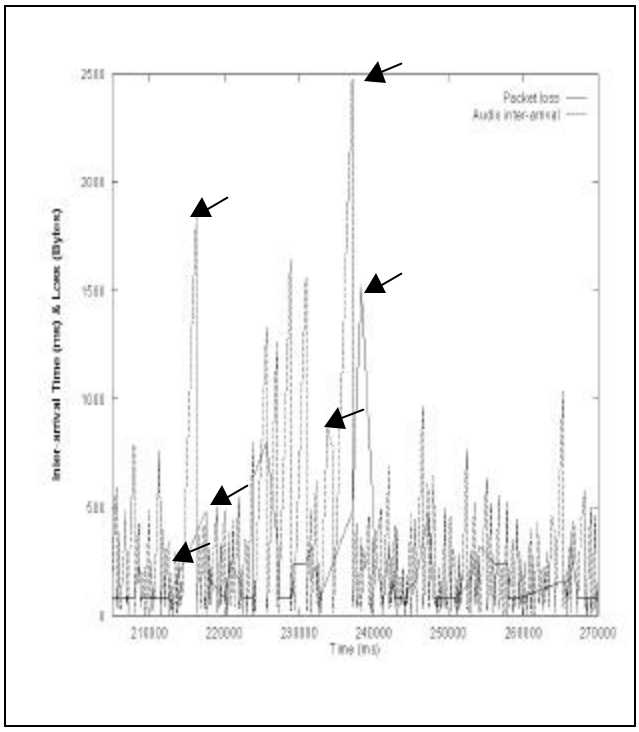


Figure 4: Audio inter-arrival time and packet loss

Figure 5 shows a plot of the relationship between the inter-arrival jitter and packet loss. Figure 6 shows an extension of the relationship in the time domain. It is observed that the negative audio jitter peaks are of larger amplitude and occur more frequently. The reason is that the inter-arrival time between packets at the receiver is often larger than that on the source since the source is usually controlled by a constant bit-rate regulator. It is also observed that the jitter at the receiver is of positive value. The reason is that the packet inter-arrival time at the receiver can be lower than that of the source. Since the inter-arrival value at the sender is usually maintained at a constant value (in our case 120 ms, since we pack four frames per audio packet), the peak positive value of the jitter is not expected to exceed this value. There may be a few exceptions to this rule as shown in Figure 5. One exception to this rule is when an error occurs at the source, causing it to violate the constant bit-rate rule. Another exception is in the case of silence periods at the source where packet inter-arrival time may be longer. These exceptions explain the very few occurrences of positive high values for the jitter.

Figure 5 shows small overshoots for packet loss following abrupt rises in the audio jitter of previous audio packets. Figure 6 shows increases for packet loss overshoots as the jitter magnitude increases for the previous audio packets. Both curves showed consistent results with packet loss moving up and down based on the amount of jitter of the previous packets. More experiments are needed in order to make a generalization for the loss-jitter relationship, yet the obtained results provide a good basis for the introduction of a loss prediction algorithm. Our observation from Figures 5 and 6 suggests that abrupt rises in jitter value may result in packet losses after a short interval of time. The relationship between audio packet loss and jitter implies that jitter computation at the receiver can predict the possibility of packet loss in the near future. Hence, by carefully monitoring inter-arrival time at the receiver, it is possible to reduce packet loss by warning the sender.

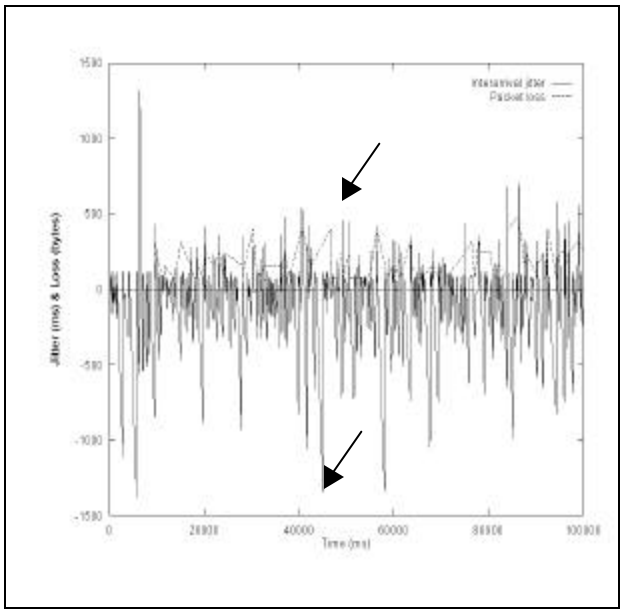


Figure 5: Inter-arrival jitter and packet loss

H.323 defines procedures for controlling the bit-rate of the multimedia streams in the conference using the H.245 [10] Flow Control command. The Flow Control command can modify the bit-rate of the channel stream or of the whole multiplex of streams. A terminal may send this command to restrict the bit-rate sent by the far-end terminal. According to H.323 recommendations, a terminal that receives this command shall comply with it. Using the H.245 Flow Control command and the relationship between audio jitter and loss, we propose an algorithm for audio loss prediction and control as outlined in the following section.

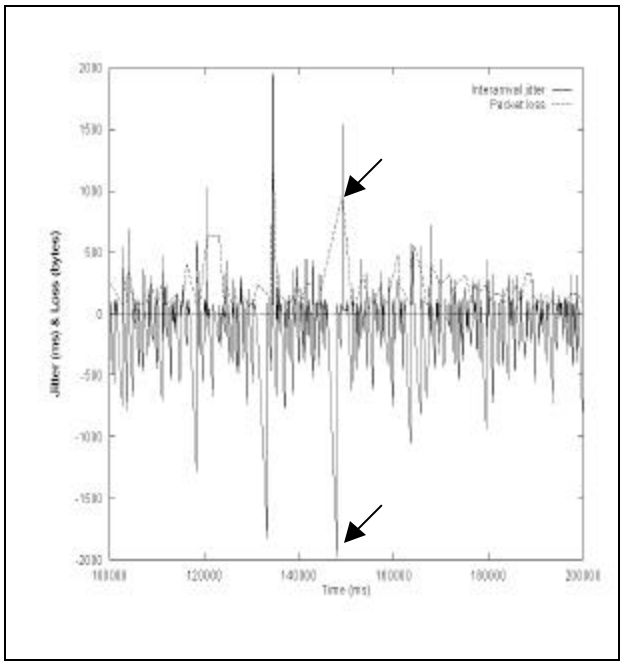


Figure 6: Inter-arrival jitter and packet loss

THE LOSS PREDICTION ALGORITHM

The previous measurements were performed for audio inter-arrival jitter and loss only, and it is yet to be seen if the same relationship applies to other real-time media components such as video. The audio loss-jitter relationship motivated the development of another algorithm that can predict the occurrence of packet loss. This algorithm works as follows:

- Compute packet inter-arrival jitter J_i for each audio packet *i* at the receiver.
- If the value of J_i exceeds a maximum value J_{max}, the network is in the congested state and loss may occur. Send a Flow Control command to the sender to limit the video bit-rate.
- If the value of J_i drops below a minimum value J_{min}, the network is unloaded, and the video bit-rate can be restored to default value. Send a Flow Control command to the sender to increase the video bit-rate.
- If the value of J_i falls between J_{max} and J_{min}, the video bit-rate will not be adjusted.

This algorithm controls the sender based on the loss predictability of the receiver. The algorithm is considered more binding than the RTCP-based algorithms since it enforces a certain behavior (increase or decrease of video bit-rate) instead of just providing feedback on the connection.

A simulation was run using the data obtained from the audio jitter-loss relationship experiment to compute the number of jitter warnings and the corresponding loss occurrences. The purpose of this simulation was to verify the loss prediction algorithm described above.

Figure 7 shows the number of jitter warnings and the percentage of unpredicted loss on the *y* axis both as a function in J_{max} (on the *x* axis). The jitter warnings are normalized against the maximum number of warnings that occurred during the experiments, and J_{max} is measured in units of the standard deviation of the inter-arrival jitter.

Using a high J_{max} value, the loss prediction algorithm was capable of predicting almost 96% of lost audio packets. As the threshold value, J_{max}, decreases, the packet loss value starts to rise.

It is observed from Figure 7 that a good suggested value for J_{max} that provides a compromise between the number of generated jitter warnings and packet loss is approximately 1.8 * jitter standard deviation. This J_{max} value provides prediction closure for a majority of packet loss.

It is possible to optimize the number of generated Flow Control commands by following certain heuristics. One possible heuristic is to limit the number of generated jitter Flow Control packets within a certain interval of time provided that the last computed jitter was not preempted. Preemption means that a new jitter value was computed with a larger absolute value than the last computed jitter value that caused the video Flow Control command to occur. Likewise, stabilization in the jitter computation at the receiver can prompt the receiver to issue Flow Control commands to raise the video flow rate.

This algorithm is more responsive than the RTCP-based feedback algorithm. It is also possible to implement using standard protocol messages such as the H.245 Flow Control command. The algorithm is also more efficient than the RTCP algorithm since flow control is generated on demand as opposed to periodically generating large statistics' packets. Further, this algorithm is based on local feedback and remote control (LFRC) as opposed to remote feedback and local control (RFLC), for example, RTCP-based algorithms. LFRC is more effective when terminals from different manufacturers are interoperating, since remote control is more binding than remote feedback. The reason is that terminals are required to comply to commands issued from other terminals but can ignore indications (such as RTCP reports). To be more specific, terminals can ignore RTCP reports and choose to not adjust their flow rate based on these reports, while they are obliged to comply with a command such as the H.245 Flow Control. In this way a terminal can effectively

control the quality of the stream and the rate of the traffic.

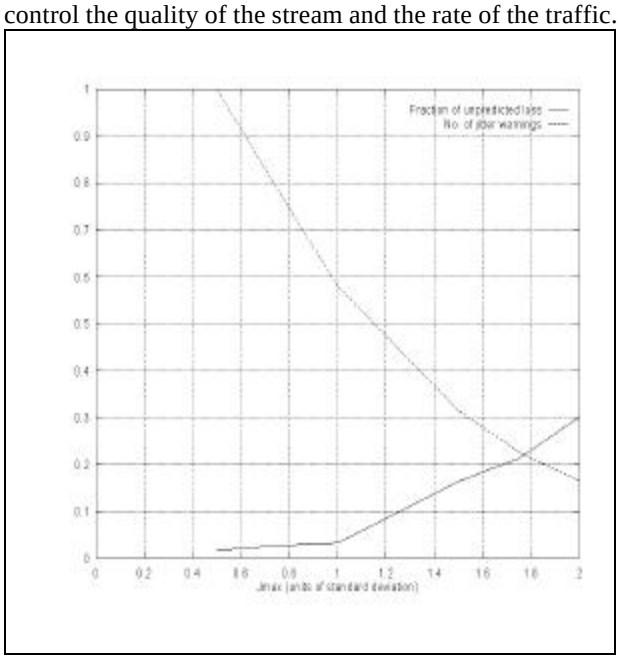


Figure 7: Loss, jitter indications, and J_{max}

CONCLUSION

Multimedia conferencing applications can be rate controlled over the Internet by continuously providing feedback about its performance over the network. Typical useful statistics include packet loss, delay, jitter, and timestamps for media synchronization. RTCP is an associated protocol with RTP designed to handle the delivery of the monitoring service of the RTP protocol. It provides periodic feedback on the quality of media packets including timing information.

We introduced an RTCP-based feedback algorithm that recovers from audio loss by controlling video bit-rate. The algorithm adapted well to network conditions; however, the response was rather slow and tardy since the feedback cycle is larger than six seconds.

We presented the limitation of RTCP-based algorithms for bit-rate control. We then attempted to discover a relationship between audio loss and packet inter-arrival jitter. We computed the jitter and the packet loss at the receiver and plotted their relationship with time. It was discovered that most of the significant packet loss was preceded by abrupt changes in packet inter-arrival time or jitter. We proposed a new algorithm that can predict audio loss based on jitter computation at the receiver. We evaluated the performance of the algorithm to assess its effectiveness. The loss prediction algorithm is more responsive than RTCP-based feedback algorithms. In addition, the prediction algorithm is enforcing control as opposed to RTCP-based algorithms that only provide feedback indications.

REFERENCES

[1] Aravind R., "Image and Video Coding Standards," *AT&T Technical Journal, Vol. 72 No. 1*, January/February 1993, pp. 67-89.

[2] Ballart R., "SONET: Now It's the Standard Optical Network," *IEEE Communications Magazine*, pp. 8-15, March 1989.

[3] Bernet Y., et al, "A Framework for Differentiated Services," *draft-ietf-diffserv-framework-02.txt*, February, 1999.

[4] Bolot J., Vega-Garcia A., "The Case for FEC-Based Error Control for Packet Audio in the Internet," in *ACM Multimedia Systems*, 1998.

[5] Burke C., Busch J., *How to Build an Internet Service Company*, systems 2000+ Publishing, 1996.

[6] Busse I., Deffner B., Schulzrinne H., "Dynamic QoS Control of Multimedia Applications based on RTP," *Computer Communications*, January, 1996.

[7] CCITT, "Recommendations on BISDN," *Geneve*, January 1990.

[8] ElGebaly, H., Toga, J., "Demystifying Multimedia Conferencing Over the Internet using H.323 Standard," *Intel Technology Journal*, Q2, 1998.

[9] ITU-T Recommendation H.323 (1996), "Terminal for Low Bit-Rate Multimedia Communication Over Non-Guaranteed Bandwidth Networks."

[10] ITU-T Recommendation H245 (1995), "Control Protocol for Multimedia Communication."

[11] Jacobson V., "Congestion Avoidance and Control," *ACM Computer Communication Review, Vol. 18*, pp. 314-329, August, 1988.

[12] Le boudec J-Y., "The Asynchronous Transfer Mode: A Tutorial," *Computer Networks and ISDN Systems, No. 24*, pp. 279-309, 1992.

[13] Schulzrinne H., Casner S., Frederick R., and Jacobson V., "RTP: A Transport Protocol for Real-Time Applications," *RFC 1889*, January, 1996.

[14] Shang, L., Deering, S., Estrin, D., Shenker, S., "RSVP – A New Resource ReSerVation Protocol," *IEEE Network Magazine*, September, 1993.

AUTHOR'S BIOGRAPHY

Hani ElGebaly is a senior software architect with Intel Architecture Labs located in Oregon. He is the technical lead within the Broadband and IP Telephony lab for the voice over IP protocol development team. His current focus is on multimedia conferencing protocol development that complies with the International Telecommunication Union standards such as H.323.

Hani received a Ph.D. degree in computer science from the University of Victoria, Canada; an M.Sc. degree in computer science from the University of Saskatchewan, Canada; and a B.Sc. degree with high honors in electrical engineering from Cairo University, Egypt. Hani has contributed to multiple international standards and profiles for the telecommunication industry through the International Telecommunication Union (ITU) and other telecommunication standards' bodies.