




Multimedia Conferencing in the Etherphone Environment

Harrick M. Vin, Polle T. Zellweger, Daniel C. Swinehart, and P. Venkat Rangan
Xerox Palo Alto Research Center



**The latest extension of
the Etherphone project
creates a powerful
conferencing system
that lets users control
their participation in
multiple conferences
across multimedia
networks.**

Traditionally, voice, video, and data communications have been handled by different communication networks. The public telephone network has been the primary voice carrier. Cable TV has transmitted video. Specialized computer networks have handled data. This historical separation is due to fundamental differences in these media. Audio and video characteristics — such as sensitivity to delay, high bandwidth requirements, and tolerance of high error rates — contrast markedly with the requirements of data. Hence, transmission of audio and video has typically involved circuit switching of analog signals, whereas data has been transmitted through digital packet-switched networks.

Although these media could not be integrated physically, various efforts were made to integrate them logically. One such example is the Etherphone system,^{1,2} developed several years ago at the Xerox Palo Alto Research Center. The original system's primary goal was to integrate the user's view of telephony, other audio applications, and conventional workstation applications. It used Ethernet for audio transmission and control, but separated the two functions, transmitting audio packets over a separate Ethernet via specialized audio hardware. Separate audio hardware improved the system's reliability, performance, and flexibility, and was largely a response to the limited performance of workstations of the time.

The Etherphone system was recently extended to incorporate video. Because networks were not available to carry high-quality digital video, we continued the earlier approach of logical integration in lieu of physical integration. Analog techniques were employed for transmission and switching, while the Etherphone's original flexible control methodology, or *connection management architecture*, sustained the integrated user view.

Limitations of specialized hardware and software prevented full physical integration and further functional extensions of the Etherphone system. However, advances in communication and computer technologies now offer high bandwidth at modest cost and high-performance workstations with multimedia capabilities. For example, Sun Sparcstations include hardware for telephone-quality audio at 64 kilobits per second, and Next workstations implement CD-quality audio at 1.44

megabits per second. Workstations supporting limited-quality motion video are also beginning to appear.

This article describes the latest extension of the Etherphone system. This extension replaces the earlier specialized audio components with Sparcstation-based audio, while retaining the newer video capabilities. We chose to integrate computing with audio processing on the same hardware for three reasons:

- to make Etherphone features available to more users by employing a widely available platform;
- to ease the development of new features by using powerful development environments; and
- to obtain sufficient computing power for additional experiments, such as mixing arbitrary numbers of audio streams or providing independent volume control for each stream.

We have used the hardware enhancements to provide the following new features:

- Users can participate fully in one conference while listening passively to others, providing a flexible means of interacting in a collaborative but distributed work environment.
- A user can establish a multimedia conversation or conference in which both

Evolution of the Etherphone environment

The Etherphone system provides a flexible basis for investigating the management of telephone and video connections and of stored media within a distributed computing environment. These objectives, rather than the exploration of any particular form of media transport (transmission and switch-

ing), have determined the design approach. The goal has been to provide basic multimedia capabilities for use by a variety of workstation applications.

Figure A shows the system architecture at three stages of growth: the original Etherphone system, the Macaw extension, and the Phoenix extension. We will

describe the hardware and user-level capabilities of each stage, then explain the underlying system design that allowed these extensions to plug in with little central change.

Hardware and user capabilities.

The original Etherphone system was

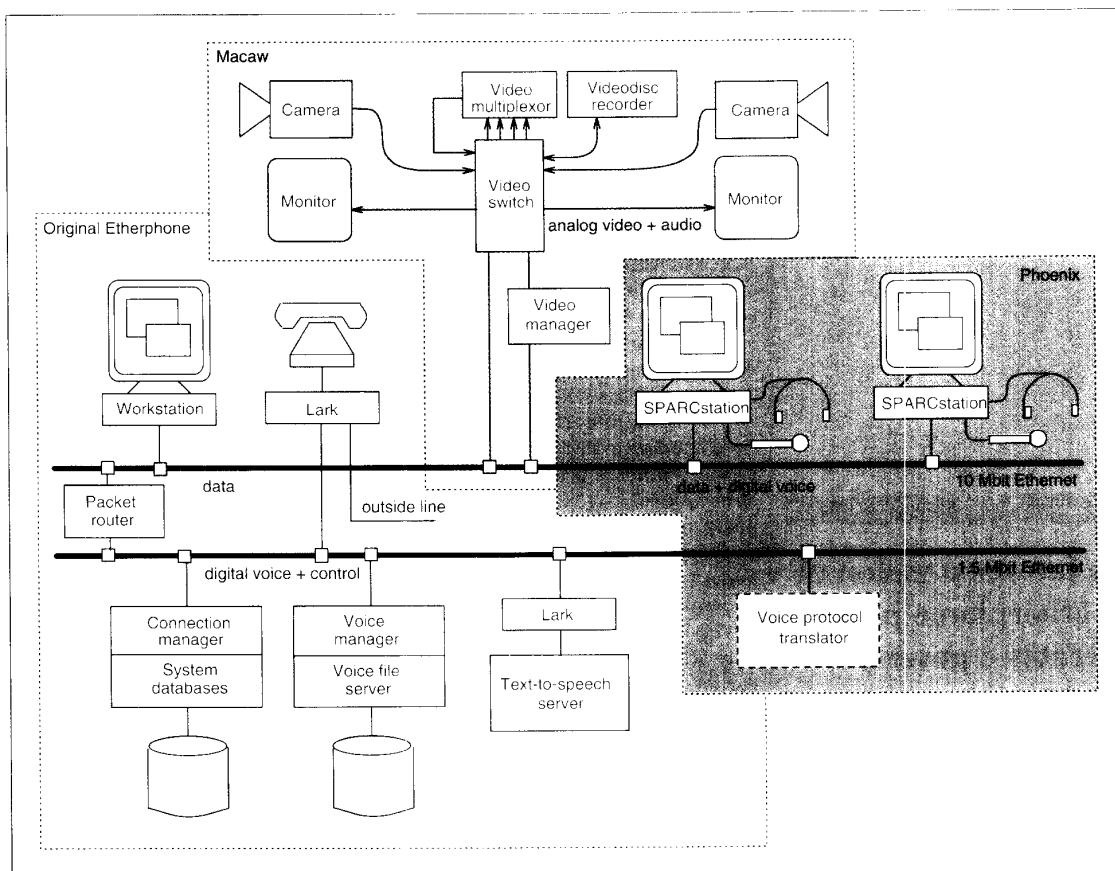


Figure A. Evolution of the Etherphone system architecture.

full and passive conferencing modes are available in audio, video, or both.

This article emphasizes the software mechanisms that support these new features: first, a Sparcstation facility called Phoenix that extends the Etherphone software architecture to permit more flexible conferencing and to control Sparcstation-based Ethernet audio transmission, and second, the integra-

tion of the Phoenix capabilities with Macaw, the earlier video extensions. We also describe our multicast packet protocol for audio transmission, which reimplements and extends the earlier special-purpose protocols, adding per-channel volume control and full support for the extended conferencing modes.

The sidebar entitled "Evolution of the Etherphone environment" describes

the genesis and growth of the Etherphone system and its underlying software architecture.

Before proceeding, we want to highlight other recent efforts to investigate the mechanisms for managing and controlling person-to-person conversations and shared text-oriented workspaces. Lantz³ proposes a text/graphics conferencing architecture whose goal is to fit into existing systems with minimal im-

constructed in the early 1980s. The custom-designed Etherphone hardware (known as a Lark) consists of a telephone/speakerphone instrument that includes a microcomputer, encryption hardware, and an Ethernet controller. Larks digitize, packetize, and encrypt telephone-quality voice (64 kilobits per second, with silence suppression), and send the voice packets to each other directly over an Ethernet. Each Lark also has a connection to a standard telephone line. Central services include a connection manager server and a voice file server, both running on a dedicated workstation, and a commercial speech synthesizer configured as a text-to-speech server. A voice manager controls the voice file server and supports sharing and editing of recorded voice via database functions. From a workstation, a Lark user can place and receive telephone calls, maintain private telephone directories, and manage a database of voice messages. An annotation package allows voice to be added to text documents (including electronic mail) and provides a simple direct-manipulation interface for editing voice. In addition, the system can "speak" text under user or program control via the text-to-speech server.

The Macaw extension replaces the audio-only system with a video/audio capability by connecting the Etherphone system to the Media Space, a preexisting network-based control system for Palo Alto Research Center's analog video/audio infrastructure. An analog video switch, controlled by a network server over an RS-232 link, routes video signals from any camera to any set of monitors. An analog *video multiplexor*, such as a four-way screen splitter, combines multiple video streams into a common view. A commercial videodisc recorder replaces the voice file server for video storage and playback; the video manager is a software database package that supports sharing and editing of recorded video. The original Etherphone connection manager continues to control and coordinate the extended system. Macaw users can place and receive

analog video/audio phone calls from their workstations. The voice annotation and editing package was extended to permit video annotation and editing as well.

The Phoenix extension replaces the specialized Lark telephones with the audio capabilities built into each Sparcstation, which can digitize audio signals at 64 kilobits per second. A Sparcstation may also have associated camera and video monitors, controlled by integrated Macaw software. Again, the resulting system is controlled and coordinated by the Etherphone connection manager, which has been modified slightly to permit the conferencing extensions. The Phoenix audio protocol is sufficiently compatible with the earlier Lark-based protocol that it would be easy to provide a protocol translator permitting Sparcstations and Larks to exchange voice. Phoenix capabilities include flexible multimedia conferencing and simultaneous participation in multiple conversations.

The Etherphone connection manager.

The Lark-based system used Ethernet packet communication for voice transport. However, anticipating the need for other transports as new technologies developed, we designed the system around a transport-independent *connection manager*. The connection manager provides the basic support for negotiation of call states among the participants in each call. It provides a common notion of state transitions representing the various phases of signaling, connection establishment and holding, and termination. It supports a common approach to reaching parties by name or number. It provides a notification service used to inform each participant of significant actions taken by the others. Finally, it provides registration and notification facilities that permit specialized agents to provide services such as audio recording, audio playback, or voice synthesis.

To achieve transport independence, the components responsible for maintaining a model of the current call state are sepa-

rated from the components responsible for implementing and controlling the individual media connections.

Within the connection manager are objects called *parties* that know nothing of the details of media, transport, or other local aspects, and therefore abstractly represent each telephone, user window, video terminal, etc., involved in a conversation. Associated with each party is an object called an *agent*, which acts as an intermediary between the connection manager and the specific implementation, converting the individual aspects of a particular transport or of a particular workstation user interface to a standard set of state-management interactions. Agents register dynamically with the connection manager, communicating with it using a remote procedure call protocol, and may thus reside anywhere within the network. Agents retain whatever state is necessary to manage both their participation in conversations and their particular physical configurations.

Two classes of agents are defined. *Transport agents* are associated with actual media terminals such as telephones. They are responsible for control interactions with the connection manager, management of the actual media, and response to local user interactions such as telephone pushbuttons. *Control agents* represent user interfaces such as workstation windows and are responsible only for connection control and user interface support. A new transport is added by creating a corresponding transport agent to represent it. Unless new functionality is also being provided, it is unnecessary to modify the connection manager or the control agent representing the workstation interface. We expect to extend the existing agent set with additional agents in the future, as new networking methods become available to provide superior transports.

pact. Sarin and Greif⁴ have also studied conferencing architectures for text and graphics only. Forsdick et al. at Bolt, Baranek, and Newman,⁵ Ahuja et al. at AT&T Bell Laboratories,⁶ and Aguilar et al. at SRI⁷ propose architectures for person-to-person audio and/or video conferencing. Casner et al. at the University of Southern California's Information Sciences Institute⁸ have developed an audio/video conferencing system integrating all media transport on the same wide-area packet network. Their research has focused on networking and protocol issues of conferencing among geographically distributed sites, rather than on flexible capabilities for user participation in conversations.

Angebrannt et al.⁹ provide a client-server architecture analogous to X Windows for integrating audio and telephony into a graphics workstation. Its emphasis is on lower level audio-resource management rather than on rich conferencing capabilities. The PX system,¹⁰ developed at Bell Northern Research's Computing Research Laboratory, is perhaps most like the Etherphone. Personal workstations cooperate with a flexible control program for a digital switch to explore workstation-controlled telephony and capabilities for editing and distributing recorded voice messages.

Phoenix conferencing capabilities

Voice and video differ fundamentally from data in their semantics and usage, and hence require different paradigms for access. In our system, media interactions among participants take the form of *conversations*, by which we mean any connection between two or more users or between users and multimedia services. A conference is a conversation involving more than two participants. The Etherphone/Macaw system, facilitated by the connection manager, permits a wide range of user actions in the management of conversations:

- *Simple conversations.* A user can make ordinary telephone calls or video phone calls, including multiparty conference calls.
- *Conversations with software services.* A user can establish a conversation with a radio service; monitor a remote

meeting; record, play, and edit recorded audio or video; or cause selected text to be spoken by a voice synthesizer server.

- *Call filtering.* Users can filter incoming calls based on urgency, subject matter, etc.

- *Dynamic membership control.* Users can dynamically join or leave conversations.

- *Call forwarding and visiting.* People do not always remain in their offices. Call forwarding automatically forwards calls to the workstation where a user is logged in. Visiting causes calls to ring simultaneously at the user's original site and at a temporary site, which is identified either by explicit command or by locator devices.

- *Background conversations.* The system can automatically place background calls, such as meeting broadcasts, on hold to accept higher priority incoming calls.

The Phoenix extensions provide the following additional features:

- *Full audio/video conferencing.* Macaw video is combined with Phoenix audio to provide full audio/video conferencing. The analog audio in the earlier Macaw system could not automatically support voice conferencing; it required manual control of an audio mixer.

- *Best-effort conferencing.* Users can participate in conferences using only audio, only video, or both. Each user can choose a different mix.

- *Multiple simultaneous conversations.* Each user may now participate in more than one conversation simultaneously. For example, while participating in a conference, a user may converse with someone else (termed a *side chat*) without disturbing the rest of the conference.

The accompanying sidebar, entitled "Phoenix user interface," demonstrates how to invoke these features.

- *Multimedia conversations.* The lack of audio-mixing hardware limited the Macaw analog audio/video conferencing. The Phoenix extensions combine analog video with digital audio (that included more flexible software mixing), and extend connection management to support conversations in a combination of media.

The ability to create conversations in multiple media gives rise to the concept of best-effort conversations. Two main approaches could be used in creating best-effort conversations:

- *Use the media supported by every participant.* The system would create a conversation in the intersection of the media set that each participant could support. The system would determine the media set dynamically based on hardware availability, forcing all participants to use the same media. If even one could not support video, an audio-only conversation would result.

- *Use the desired media.* This more general model permits asymmetric participation in conversations. For instance, a lecture may be broadcast in the audiovisual domain, but students can participate by viewing only the slides (video only), listening only to the lecture (audio only), or doing both. We have taken this more lenient approach due to its generality and flexibility.

When a participant engages in multiple conversations involving audio, the system digitally mixes all the audio streams received for each conversation. Similarly, if the conversations involve video, the system displays the video streams received for each conversation simultaneously, using a video image multiplexor that places up to four video images on the screen.

Multiple simultaneous conversations.

In real-life situations, a person generally participates in only one interaction at a time, but can passively monitor many other things while doing so. To represent this mode of interaction, the Phoenix extensions permit users to participate in multiple conversations simultaneously, providing a convenient environment for collaborations. For example, during a multiperson meeting, participants forming a subcommittee may convene privately and later report to the meeting (see Figure 1a), or a participant may consult privately with an outside party (see Figure 1b). Participants in either of these side chats continue to monitor the original meeting. The Etherphone system originally provided only simple two-way conferencing and special one-way support for monitoring meetings.

There are two models for supporting these extensions. The first permits ac-

Phoenix user interface

Figure B shows a view of Harrick Vin's telephone management window. Vin is calling his coauthors — Swinehart, PolleZ (who is currently on hold), and Venkat — about this article. During the call, he has placed a side chat call to his secretary, Fear, to resolve a timely question. Vin continues to listen to Swinehart and Venkat converse while he talks with Fear.

To place a conference call, a caller

types a list of recipients in the dialing area or selects several recipients from the conversation log. If desired, the caller can fill in a subject or cycle among several choices for the Priority and Media fields. The control menu includes conferencing features that let callers terminate ringing for recipients who have not answered, add new recipients to an ongoing call, or place side chat calls (which automatically place the old conversation

in a Recv-only state). Additional accelerators and features, such as manually toggling the Recv-only condition to mute the conversation, are available via mouse clicks on conversation log entries. The conversation log distinguishes between Active and Recv-only states for the current user only. To protect privacy, conversation logs of other participants display both states as "active."

Finch 7.0						
Phone	Answer	Disconnect	SpeakText	StopSpeech	Directory	Comment
ConfEstablished	Invite	SideChat	SetPriority			
Called Party:		Fear.pa			Calling Party: Pier.pa	
Subject:		release forms?			Priority: urgent	
Media:		Audio				
June 25, 1991 10:36:42 am PDT						
42: Placing sidechat call to Fear.pa						
42: Subject: release forms?						
24 Jun 91	4:54:41pm	completed	00:05:23	conference call from Swinehart.pa		
		idle		Priority: normal	Media: AudioVisual	
		idle		Swinehart.pa		
		idle		Vin.pa		
		idle		PolleZ.pa		
25 Jun 91	9:13:37am	abandoned	00:00:23	from Want.pa		
25 Jun 91	9:48:49am	busy	00:00:14	to Terry.pa		
25 Jun 91	10:27:12am	active	00:10:15	conference call from Vin.pa		
		active		Priority: important	Media: BestEffort	
		active		Vin.pa		
		on hold		Swinehart.pa		
		active		PolleZ.pa		
		active		Venkat.pa		
25 Jun 91	10:36:42am	active	00:00:45	to Fear.pa		
		Subject: release forms?	Priority: urgent	Media: Audio		

Figure B. Workstation telephone management window. The Phoenix user interface includes (from top to bottom) a control menu, a dialing area, a system status area, and a conversation log. The log shows a permanent history of this user's calls, including the current state of each conference participant. In this example, the last two calls are still in progress.

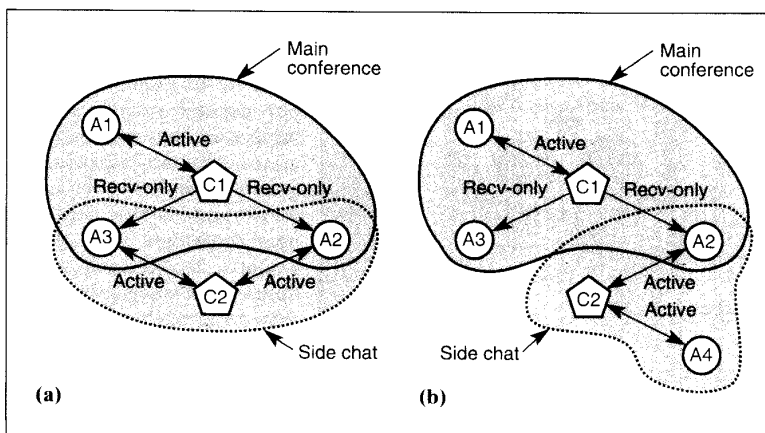


Figure 1. Side chat within a conference: (a) Users A₂ and A₃, participating in C₁, are engaged in a side chat C₂ while also listening to the main conference C₁; (b) A₂ is conversing with A₄ (in C₂) while also listening to C₁. None of the other participants of C₁ can hear the side chat.

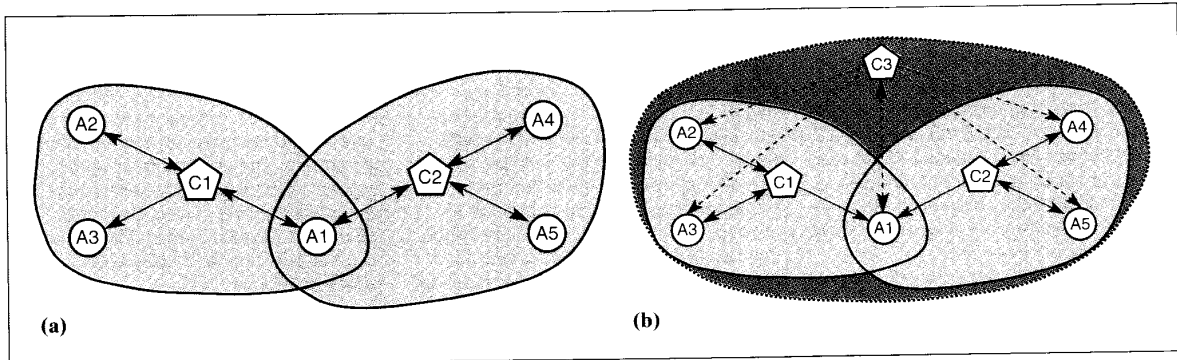


Figure 2. Active participation in multiple conferences. In (a), a participant is sending to more than one conversation simultaneously (to-many, from-many). In (b), the example in (a) is reduced to a to-one, from-many case by introducing dummy conversation C₃. Arrows show the direction of transmission in each conversation.

tive participation in multiple conversations, yielding a (*transmit*)-to-many, (*receive*)-from-many model of collaboration. The second permits active participation in only one conversation with passive listening to any others (a to-one, from-many model). Phoenix conferencing employs the to-one, from-many model because it captures most collaborative scenarios and is straightforward to implement. It is also functionally complete, since to-many, from-many scenarios can be reduced to a combination of to-one, from-many conversations, as shown in Figure 2. In fact, both these models can also be implemented by a composition of one-

to-many simplex communication channels, such as those described by Crutcher and Grinham.¹¹

Software architecture

This section describes the major software components and activities of the Phoenix version of the Etherphone system. Figure 3 shows the following components:

- *The connection manager.* Cooperates with transport agents and control agents to control the progress of multimedia conversations. (The earlier side-

bar on the evolution of Etherphone discusses the underlying system architecture.)

- *The Finch control agent.* A user workstation module that provides a window interface for placing, joining, and terminating conversations. It also maintains a persistent conversation log.

- *The media transport agent.* Controls media transmission. Two modules provide its functionality: PhoenixAudio, which implements the multicast audio packet protocol, and MacawVideo, which communicates with the video server to control the switching of analog video and audio. A top-level software module works with the connection manager to coordinate the activities of the two media-specific modules.

- *The multicast server.* Helps implement the Phoenix audio protocol.

- *The video server.* Connects MacawVideo to the analog video switching infrastructure.

The multicast server and video server are examples of transport-dependent services. These services are implemented as separate servers rather than as components of the individual agents for two reasons. First, each manages contention for a shared resource. Second, each can thus be used for applications other than conferencing.

Agents and servers communicate using a remote procedure call protocol. Several RPC implementations are available.

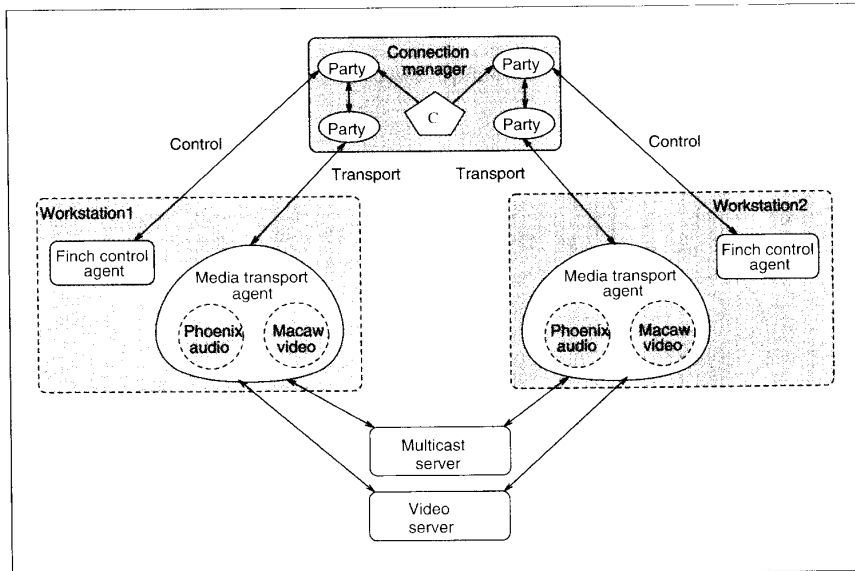


Figure 3. Phoenix software architecture.

but all of the agents described in this article use Sun RPC.

Transport-dependent servers. The multicast server supports PhoenixAudio. When there are more than two participants, internetwork packet-multicasting techniques are used to avoid transmitting more copies of each audio packet than necessary. The multicast server has two management functions to centrally manage the address space for the multicast groups supporting these conferences: GetMulticastAddress() creates a new multicast group; FreeMulticastAddress() frees one.

The video server is the interface between MacawVideo and the Media Space. PARC's existing analog video/audio infrastructure. The video server provides a channel allocation function for analog media and translates MacawVideo requests into sequences of simple commands to control the video switch and related peripherals. ReserveMultiplexor() reserves a video multiplexor for a conversation; FreeMultiplexor() makes one available for reuse. ConnectCamera() routes video originating from a specified camera or other video source to a multiplexor input port. Lastly, ConnectMonitor() routes video from a multiplexor output to a specified destination. The current implementation always uses the four-way video multiplexor, although the video server can support direct, full-screen connections.

Connection manager concepts. Connection management includes the following basic principles:

- *Negotiated conversation establishment.* Setting up a conversation is a negotiation between the caller's agent and the agents of the other participants, whom we call *recipients*.

- *Autonomous participation in conversations.* Every agent is autonomous in that each one can independently decide whether to participate in a conversation and when to leave it.

The connection manager provides functions to create conversations and to control the progression of agents through conversations. An agent can request the creation of a conversation by calling the function CreateConversation(), and can then invite other participants to join by calling Alert(). (The recipient party enters the state Notified, after which its

own agent controls all further activity.) An agent informs the connection manager about its state changes using Advance(), and the connection manager uses ProgressReport() to report these transitions to all other agents. Similar functions allow agents to register specialized services (for example, speech synthesis) and to be notified when these services generate distinguished events (for example, the completion of one synthesized utterance or the beginning of the next one).

When an agent A wishes to change its state in a conversation, it issues an Advance request to the connection manager. The connection manager verifies that the transition is permissible and updates the official version of the conversation state to the new value. Subsequently, agent A takes any necessary local actions to meet the requirements of the new state, while the connection manager concurrently sends ProgressReports to the other participants. Other agents can respond by merely noting the state change, or they can repeat the process by issuing their own Advance requests. The sidebar entitled "Etherephone conversation model" describes the meanings of the states that agents use to control their progress through conversations.

Functionality of the media transport agent. We now describe the media-specific actions performed by PhoenixAudio and MacawVideo to properly maintain the state of the media transport agent in a conversation.

- *Idle → Initiating.* When the caller's Finch control agent makes this transition, the caller's media agent responds. The media agent's PhoenixAudio takes no action, but its MacawVideo makes two requests of the video server: first, to route the caller's camera to one of the input ports of a video multiplexor assigned to this conversation and, second, to route the output of the multiplexor to the caller's own monitor. The appearance of the caller's camera image on the monitor is a video equivalent of a "dial tone."

- *Idle → Notified.* In the Notified state, the agents perform call-filtering decisions. No transport-specific actions are taken by PhoenixAudio or MacawVideo.

- *Notified → Ringing.* PhoenixAudio starts playing a personalized ring tune through the recipient's workstation

speaker. MacawVideo connects the caller's camera to the recipient's monitor in a one-way connection that is the video equivalent of a ringing telephone.

- *Initiating → Ringback.* If no other audio conversation exists, the caller's PhoenixAudio also starts playing the recipient's ring tune. MacawVideo takes no action. Thus, a video-only user will receive only limited feedback, in the Finch workstation window, during call setup.

- *Ringing → Active or Ringback → Active.* The media agent's PhoenixAudio directs the audio protocol implementation to begin sending and receiving packets for the connection or to mix the new party's audio into any ongoing conversation. If the transition is from Ringing, then MacawVideo requests the video server to establish a video conversation by routing its camera to the multiplexor assigned to this conversation. If the transition is from Ringback, MacawVideo takes no action. If the user is already participating in a video conversation, an additional per-user multiplexor could simultaneously display video streams being received from all the conversations. Figure 4 (on the next page) illustrates how two users establish a video conversation.

- *Active → Recv-only.* In the Recv-only state, PhoenixAudio stops transmitting audio packets to the multicast group associated with the conversation, and MacawVideo requests the video server to disconnect the user's camera from the multiplexor associated with the conversation.

- *Recv-only → Active.* PhoenixAudio resumes transmitting audio packets to the associated multicast group, and MacawVideo requests the video server to route the user's camera to the next available input port of the multiplexor assigned to the conversation.

- *Active → Inactive.* This participant is putting the conversation "on hold." PhoenixAudio terminates voice transmission and reception, and MacawVideo directs the video server to disconnect all connections to or from the assigned multiplexor. The agents maintain call information, such as the multicast group address, for use in rejoining the active conversation later.

- *Any state → Idle.* PhoenixAudio and MacawVideo terminate all connections and transmissions, then delete all information regarding the conversation from their information bases.

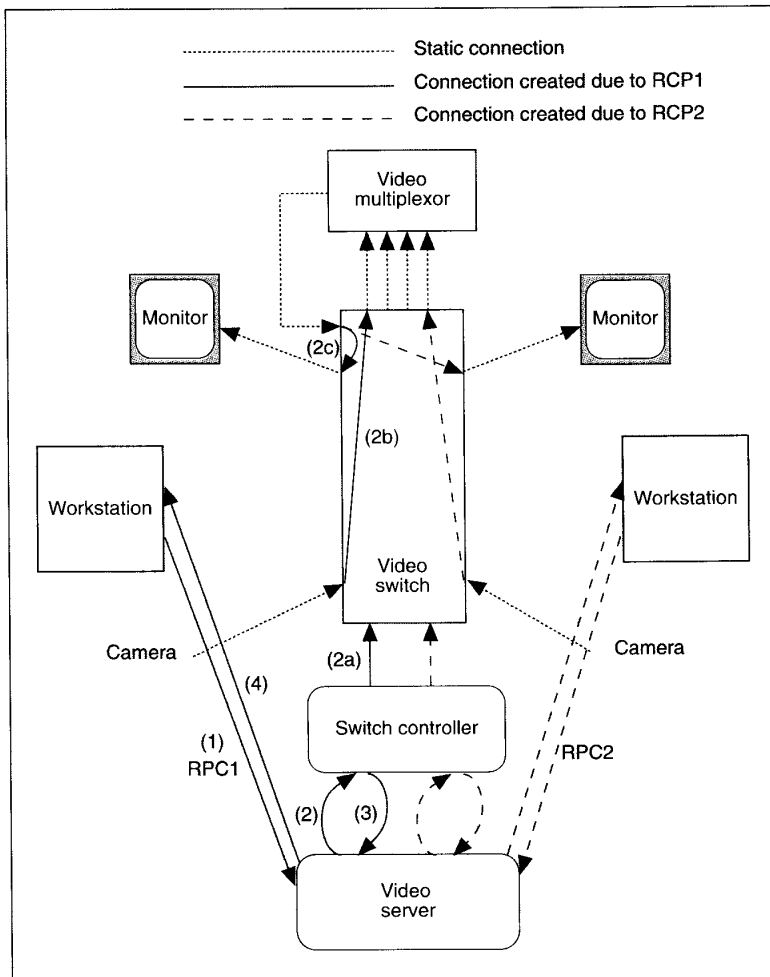


Figure 4. Establishing video connections: (1) MacawVideo asks the video server to create a video channel. (2) The video server calls a procedure in the video switch controller, a network service that directly manages the video switch connections. (2a) The switch controller executes the switching request. As a result, (2b) the video camera is connected to the multiplexor and (2c) the multiplexor is connected to the monitor. (3) The switch controller sends result information to the video server, (4) which reports it to the media agent.

Phoenix audio transmission protocol

The original Etherphone system uses Ethernet packet transmission between the specialized Lark processors. The audio protocol supports communications between Larks and between Larks and other services, including limited conferencing. This section describes the audio protocol, derived from the Lark protocol, that we developed for the Sparcstation.

Once the system establishes audio connections for a conversation, audio communication can take place among the participants. Audio communication between users is a real-time problem. Audio transmission is distinguished by the following characteristics:

- The perceived end-to-end delay must be constant and small (ideally, below 50 milliseconds).
- In ordinary conversations, people usually take turns speaking. Furthermore, speech alternates "talk spurts"

with silence. Averaged over many voice conversations, less than 50 percent of the full-duplex channel capacity is used.

- The human ear can tolerate brief distortions in speech. Thus, occasional transient errors in the digital representation of speech can be ignored. Similarly, dropouts of up to several milliseconds are perceived as pops and clicks, and are tolerated.

These characteristics, the most critical of which is the delay requirement, must be accounted for when designing a datagram-based audio transmission protocol.

In Phoenix, each participant transmits audio packets to the multicast group assigned to the conversation. Every participant receives packets transmitted by every other participant and then digitally mixes them before playback. This approach permits each participant to determine which of the received audio streams to include in the result, and at what volume.

Figure 5 shows the format of audio protocol packets. Each packet contains up to 160 mu-law-encoded bytes (20 milliseconds of audio, sampled at 8,000 bytes per second). Thus, at most, 50 audio packets are transmitted on the network every second for each call participant.

Before transmitting an audio packet, the sender computes an "energy" value for it. The energy value is related to its volume level. To perform silence suppression, the protocol defines an "energy threshold" below which it asserts that the packet is silent and, hence, does not transmit the packet. However, to achieve smooth sound-to-silence transitions, the sender continues to transmit packets with energy below the threshold until a certain number of such packets have been transmitted. In addition, the amplitude of sound in these packets is attenuated by a quadratic reduction function. This attenuation technique was first described to us by D. Sincoskie and J. DeTreville, who were then at AT&T Bell Laboratories, where it was used in their Packetphones.

At the beginning of a new talk spurt, the sender prepends 10 milliseconds of sound from the previous "silent" packet and then transmits the larger packet. This provides a smooth silence-to-sound transition by capturing the audible but low-energy beginnings of each utterance. The amount of audio captured

Etherphone conversation model

Each agent participates in negotiations to establish conversations. In this process, the agent progresses through the sequence of states shown in Figure C. The meaning of each state in this transition diagram is as follows:

- Error*. An audible error notification, such as a busy signal, is being given to the user. The user must take explicit action to terminate this condition.

- Idle*. An agent is no longer participating or has never participated in this conversation.

- Initiating*. The caller's agent enters this state in the process of alerting a recipient's agent of an attempt to establish a conversation.

- Notified*. A recipient's agent is being alerted about an attempt to establish a conversation. This agent then uses call-filtering information specified by the user to determine its next state. For example, if a user has declared calls from a caller as "hotline" calls, then when alerted by that caller, the user's agent directly enters the Active state. Otherwise, the recipient's agent normally enters the Ringing state, but

might instead decide, based on other user specifications, to reject the call by returning to Idle.

- Ringing*. When receiving an attempt to establish a conversation, a recipient's agent enters the Ringing state from the Notified state, indicating that an attempt is being made to alert a user.

- Ringback*. The caller's agent enters Ringback upon learning that the recipient's agent has entered Ringing, indicating to the caller that the recipient is being alerted.

- Active*. This is the receive-transmit state. All Active agents have bidirectional channels established among them. The recipient's agent enters Active from Ringing as soon as the recipient or a programmed agent has indicated an intent to do so; the caller's agent enters Active after learning that the recipient has answered.

- Inactive*. This is the "hold" state. An agent enters Inactive to place a conversation on hold. Thereafter, it transmits or receives no information for this conversation. In multiparticipant calls, the agents of other Active participants may continue to converse.

- Recv-only*. In this state, an agent does not transmit in any media to a conversation. However, the agent continues to receive information from the other participants.

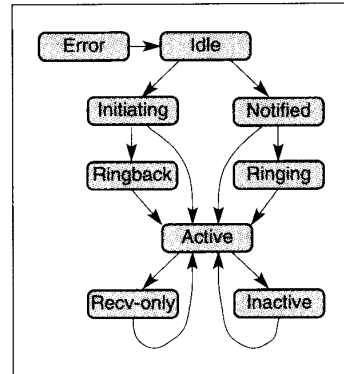


Figure C. State diagram for conversation model. In addition to the transitions shown, there is an implicit path from all the other states to the Idle and Error states. Initiating and Notified are parallel states

from the previous packet also determines the anti-jitter delay that is introduced at the beginning of each talk spurt. This delay provides the buffering necessary to accommodate network delay variations.

The receiver coordinates its local time with the receipt of the first packet of a talk spurt and uses the coordinated real time to predict the scheduled playback time of packets from that sender. The packetNumber and transmissionTime fields in the packet header are used to enforce the ordering of received packets and to correct for differences in the clock rates of sender and receiver. If the sender's clock rate is slower than the receiver's clock rate, then the sender's packets will arrive at the receiver after the receiver's clock has passed its predicted playback time. Rather than dropping all such packets, the receiver periodically resynchronizes with the sender's packets (after receiving a fixed number of late packets) by incrementing the predicted playback time by one packet. Similarly, the protocol detects packets arriving too early. When a packet is received with a scheduled playback time

too far in the future, the receiver synchronizes with the sender's stream by dropping all the buffered packets and then playing this "too early" packet. Both procedures result in noticeable click-like interruptions, which are tolerable if they are not too frequent. The sending and receiving clocks must agree well enough for the typical talk spurt to be shorter than the time it takes the receive buffers to underflow or over-

flow. It is easy to achieve this accuracy with current crystals.

Performance evaluation

The Phoenix extensions have been in operation at Xerox PARC for several months, and the system's performance

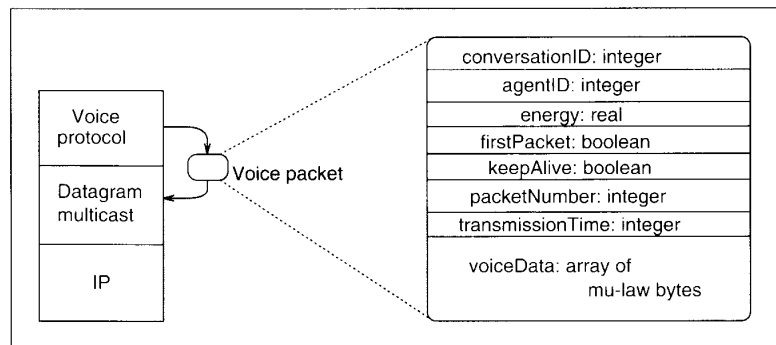


Figure 5. Protocol stack and packet format.

Table 1. Typical performance measurements for connection management operations.

Connection Management Actions	Time (in milliseconds)		
	Audio Domain	Video Domain	Audio +Video
Delay between caller initiating a conversation and recipient being alerted	100	172	172
Delay between caller initiating a conversation and receiving ringback	180	252	252
Delay between recipient clicking Answer and caller going to active state	100	172	172

Table 2. Performance measurements for voice protocol.

Factors	Time (in milliseconds)
End-to-end delay for voice samples	50
Processing delay for each packet at the sender	4
Time required to mix and adjust volume of two audio packets	3

has been within the tolerable limits of commercial telephones. This section describes performance measurements to validate this claim.

Table 1 summarizes some notable state transition times for our conferencing system. Note that the transition times of an agent with respect to a conversation are dependent on the media, since different techniques and hardware are used to perform state-specific actions in the audio and video domains. In evaluating these performance measurements, be aware that a Sun-to-Sun RPC averages 10 milliseconds, the hardware response time of the video switch controlled by a Sun workstation averages 62 milliseconds, connection manager operations such as Advance take about 20 milliseconds, and a ProgressReport reaches other agents in about 80 milliseconds. Multiple processes in the connection manager allow sending ProgressReports to multiple agents in parallel.

In the audio domain, delays result purely from the overhead of communicating state transition information. The

first and third rows in Table 1 sum the times required for an agent to receive a ProgressReport and to Advance its state in response. The second row also includes the time required for a ProgressReport, resulting from the recipient's Advance, to reach the caller. Establishing a conversation involving video requires communication with the video server (hence, with the video switch controller), which causes additional delays.

Table 2 summarizes the most salient performance measurements of the voice protocol. The end-to-end delay is the sum of the packetization delay (20 milliseconds), the transmission delay (including sender processing delay, packet transmission, reception and receiver processing time), and the anti-jitter delay (10 milliseconds). The sender's processing overhead includes the voice energy computation, silence detection, and other functions performed for smooth silence-to-sound transition.

Future work

We have identified areas for further work, including access control, a lightweight connection management model, and media synchronization.

The Etherphone connection management should be enhanced to support different types of conversations. For example, in classroom conversations, each user may have different participation (or access) rights. These rights determine the type of media transmission

channels that must be created for that conversation. For example, if a participant only has permission to receive, then it is necessary to create a simplex communication channel. It is possible to simulate a simplex channel by discarding packets received from a participant with no transmission rights. However, to actively prevent such a participant from sending requires lower level access control (for example, multicast group membership controls).

We believe that future applications such as video browsing will require a lightweight approach to establishing and managing connections. An interesting research issue is the integration of such lightweight applications with our current, more deliberate, negotiation-based models.

Since our system transmits digital audio and analog video, the corresponding delays can vary. The current implementation does not yet provide explicit means to synchronize these media streams. Media synchronization is receiving attention from several researchers, among them Nicolaou.¹²

The Phoenix extensions to the Etherphone system are a step toward integrated multimedia conferencing. They are built on Sparcstations, widely available platforms that can support powerful user interfaces, direct digital conversion and transmission of audio, and advanced telephone-like control functions. Video is transmitted over a separate cable TV network. The overall system achieves physical integration of audio and logical integration of video.

The enhanced conferencing paradigm extends Etherphone telephone conferencing to let users participate in multiple conversations employing multiple media. The Phoenix extensions support a collaboration model in which users can participate actively in one conversation while participating passively in others. The model permits ready extension to active participation in multiple conversations. In addition, the Phoenix control methods permit asymmetric conversation participation (with respect to media used).

We believe that the experience gained from building the Macaw and Phoenix extensions will prove valuable in the future design of completely digital multimedia conferencing systems. ■

Acknowledgments

P. Venkat Rangan was supported by NSF Research Initiation Award No. NCR-9009757.

We thank the referees and Eve Schooler for valuable comments on this article. We also thank Molly Mackinlay for her patience.

References

1. D.C. Swinehart, "The Connection Architecture of the Etherphone System," Tech. Report CSL 91-8, Xerox PARC, Palo Alto, Calif., 1991.
2. P.T. Zellweger, D.B. Terry, and D.C. Swinehart, "An Overview of the Etherphone System and its Applications," *Proc. Second IEEE Conf. Computer Workstations*, IEEE, New York, 1988, pp. 160-168.
3. K.A. Lantz, "An Experiment in Integrated Multimedia Conferencing," *Proc. ACM Conf. Computer-Supported Cooperative Work*, ACM, New York, 1986, pp. 267-275.
4. S. Sarin and I. Greif, "Computer-Based Real-Time Conferencing Systems," *Computer*, Vol. 18, No. 10, Oct. 1985, pp. 33-45.
5. H.C. Forsdick, "Explorations in Real-Time Multimedia Conferencing," *Proc. Second Int'l Symp. Computer Message Systems*, IFIP, Geneva, Sept. 1985, pp. 331-347.
6. S.R. Ahuja, J.R. Ensor, and D.N. Horn, "The Rapport Multimedia Conferencing System," *Proc. ACM Conf. Office Information Systems*, ACM, New York, 1988, pp. 1-8.
7. L. Aguilar et al., "Architecture for a Multimedia Teleconferencing System," *Proc. ACM SIGcomm 86 Symp. Comm. Architectures and Protocols*, ACM, New York, 1986, pp. 126-136.
8. S. Casner, K. Seo, W. Edmond, and C. Topolcic, "N-Way Conferencing with Packet Video," Tech Report ISI/RS-90-252, USC/Information Sciences Institute, Marina del Rey, Calif., 1990.
9. S. Angebrannt et al., "Integrating Audio and Telephony in a Distributed Workstation Environment," *Proc. Summer 1991 Usenix Conf.*, 1991, pp. 419-436.
10. R. Kamel, K. Emami, and R. Eckert, "PX: Supporting Voice in Workstations," *Computer*, Vol. 23, No. 8, Aug. 1990, pp. 73-80.

11. L. Crutcher and J. Grinham, "Connection Management for an Integrated-Services Network and its Application to the Multimedia Communications of a Distributed Team," Tech. Report TR-90-062, Int'l Computer Sciences Institute, Berkeley, Calif., 1990.
12. C. Nicolaou, "An Architecture for Real-Time Multimedia Communication Systems," *IEEE J. Selected Areas of Comm.*, IEEE, New York, Vol. 8, No. 3, Apr. 1990, pp. 391-400.



Harrick M. Vin is a doctoral candidate in the Department of Computer Science and Engineering at the University of California, San Diego. His research interests are multimedia computer systems and ultra high speed networking with emphasis on conferencing and storage architectures for digital video and audio. He is a recipient of an IBM Doctoral Fellowship and an NCR Innovation Award, and was a research intern at Xerox PARC in 1990.

Vin received his BTech in computer science and engineering from the Indian Institute of Technology, Bombay, India, in 1987, and his MS from Colorado State University in 1988. He is a member of the IEEE Computer Society.



Polle T. Zellweger joined Xerox PARC in 1981 and has been active in the Etherphone project since 1985. She is currently manager of the Active Documents Group in the Elec-

tronic Documents Laboratory. Her research interests include human-computer interaction, multimedia systems, and documents that contain dynamic behavior, such as hyperlinks, audio, video, and computations.

Zellweger received a BA in mathematics from the University of Alaska, an MS in computing and information science from the University of New Mexico, and a PhD in computer science from the University of California at Berkeley. She is a member of the IEEE Computer Society.

Daniel C. Swinehart is a principal scientist at Xerox PARC, performing research in high-speed communications and connection-management methodologies in support of future multimedia applications. His research interests have also included distributed systems, graphical user interfaces and editors, programming environments, and laser-printer software architectures. He is the principal architect of the Etherphone system.

Swinehart received his PhD in computer science from Stanford University in 1974. He is a member of the IEEE Computer Society.



P. Venkat Rangan directs the Multimedia Laboratory at the University of California, San Diego, where he is an assistant professor of computer science. He is also a visiting scientist at Xerox PARC. His research interests are multimedia computer systems and networks. Rangan has co-authored more than 15 publications, and has been a recipient of NSF research initiation and NCR innovation awards.

Rangan received his BTech degree from the Indian Institute of Technology, Madras, India, where he was awarded the President of India Gold Medal in 1984. He received his PhD in computer science from the University of California at Berkeley in 1988. He is a member of the IEEE Computer Society.

Readers can write to Harrick Vin and Venkat Rangan at the Dept. of Computer Science and Eng., Mail Code C-0114, UC San Diego, La Jolla, CA 92093. E-mail addresses for the authors are vin%cs@ucsd.edu., Zellweger.parc@xerox.com, Swinehart.parc@xerox.com, venkat%cs@ucsd.edu.