MAS.632 Conversational Computer Systems
Fall 2008

11

Telephones and Computers

The previous chapter discussed interactions between people, telephones, the central office switch, and the remainder of the telephone network. In this chapter we consider the benefits computers offer for call management through the enhanced services and flexible call routing they can provide. Computers can provide a user interface more powerful than the keypad or low-power display of conventional telephones. Computers can personalize telecommunications and change the way we think about using telephones.

The signaling and transmission functions of the telephone were explained in the previous chapter. Extending signaling capabilities to computers allows the computer to place calls, employ a variety of means to alert the user to an incoming call, and automatically route calls on the basis of who is calling, the time of day, and the current activity of the called party. Interfacing computers to the transmission phase of the telephone network allows the user to place calls by name or from other personal information management applications. Computers can be sophisticated answering machines providing store-and-forward capabilities so that we can "converse" asynchronously; the telephone may prove to be the primary source of voice as data to the office workstation.

Current digital telephone switches and voice mail systems are in fact computers, some running operating systems similar to those on our workstations. But the programs and run-time environments of these switches are specialized for telephone tasks, such as switching voice circuits in real time, maintaining billing records, and periodically running diagnostic checks on the integrity of the network and its constituent circuits. Telephone switches are designed to execute a specific task quickly and reliably; they are not designed to run user applications.

This chapter examines telephony applications that run on the same workstations we use to edit documents, develop software, or read our electronic mail.

This chapter begins by exploring the benefits of involving a workstation in the call management process and then offers examples of the kinds of services it can provide. The hardware requirements and underlying architectures that allow workstations to participate in call setup are then discussed. Finally, several case studies are presented for a more detailed description of the services that can be offered by interfacing computers to telephones.

MOTIVATION

In many contemporary office environments it is common to find a workstation or personal computer and a telephone in each office, a communal fax machine, and a voice mail system housed in a telecommunications closet. The voice mail system interacts with the telephone via a message-waiting light but otherwise these devices each function in isolation. This chapter proposes linking the functionality of these various devices with particular emphasis on including telephony functions in the computer, from which a number of benefits may be realized.

Access to Multiple Communication Channels

Providing *ease of access* to communication facilities over different media motivates computer-mediated telephony. Facsimile use has exploded, fostering boards that allow personal computers to send and receive faxes as well as display bit-mapped images on the screen. Many users keep their phone number databases on computers and consult them when placing a call, or dial by holding the handset next to their palm-top computer. Electronic mail use is taking off and is now widely available; its existence relies on the computer network, but messages are also being forwarded to pagers and cellular telephone-equipped mobile computers.

Making all these communication media available from a single desktop computer allows the user to easily cross media boundaries during a single communications session. For example, receipt of an email message might imply that the difficult-to-reach sender is in the office and hence available by phone for a more interactive discussion. Then during the course of discussion, conversants could exchange pertinent documents by email or fax. A caller may prefer to send a message by email instead of leaving one with a receptionist if the called party is unavailable.

Integrating disparate communication media at the workstation affords immediacy and convenience to employing multiple media simultaneously or sequentially. Bringing incoming messages into a single computing entity simplifies the task of discovering whether any new messages have arrived and enables a variety of alerting, filtering, or message forwarding policies under control of the recipient.

Improved User Interfaces

A second reason to integrate telephony into the workstation is the potential for significantly more powerful user interfaces than can be provided with telephone equipment alone. Most sophisticated telephones provide a touchtone keypad and perhaps a few dedicated function buttons such as *hold, conference,* and *transfer.* User interfaces to all telephony services must be implemented using these buttons, and feedback to the user is limited to a small repertoire of sounds such as the busy or dial tone signals. Without a display, it is difficult for the user to determine the current state of any features that may have been activated. For example, my telephone has speed dialing but it cannot tell me the contents of my speed-dialing list; instead, I have to place the call and see who answers or tape a piece of paper with my speed-dialing list to the phone. Likewise, I can touch a *redial* key to dial the most recently called number but cannot find out what that number is without trying it. I can key "*73" to cancel my call forwarding, but I cannot find out the number to which my calls are currently being forwarded.

Each year a greater number of sophisticated telephone services are offered: call waiting, call back the number that last phoned, block calls from certain calling numbers, assign distinctive ring to some calling numbers, identify multiple incoming lines with a distinctive ring, speed dialing, calling line identification, and voice mail, to name just a few. But the telephone keypad provides a sparse user interface, making it difficult to use even the existing services much less new ones. A few users may read the manual to safely reconfigure their phone, but many avoid or underutilize the new services.

One approach to providing improved user interfaces has been to build more complex telephones, with displays, soft keys, menus, and even voice prompts. However, workstations are already suited for this task. Computer displays are typically high-resolution CRT screens, whereas telephone displays if available are small and difficult to read due to hardware cost and electric power limitations. Workstations offer additional and more elaborate input devices including full keyboards and mice. Even more beneficial is the fact that powerful window systems provide rich and flexible user interfaces and allow several concurrent applications to share the screen. Workstation operating systems allow resources such as memory or digital signal processors to be shared among a number of applications; the hardware in a "smart telephone" is wasted when the telephone is idle.

There is limited experimental evidence on the value of screen-based interfaces to telephony. [Roberts and Engelbeck 1989] compared the computer and advanced telephone user interface media using standard touchtone telephones, display-based telephone terminals, and telephones augmented with speech synthesis for voice menus. They found the display-based interface both faster and most well received by subjects who routed calls, screened calls, and retrieved messages. But long-term studies based on MICE users (Bellcore's telephony development environment, described shortly) were somewhat less convincing. [Root and Koster 1986] substituted mnemonic key codes for the conventional numeric ones (e.g., "CF" instead of "72" for call forwarding). They found that subjects recalled more

system functions with the mnemonic codes but did not increase their use of the services. Similarly, [Root and Chow 1987] found that users readily employed a screen-based interface to manipulate and edit their speed-dialing lists, but this did not increase the number of entries they kept in the lists. Although subjects could speed-dial by the screen in addition to the keypad, they did not do so. Interestingly, speed-dial usage increased somewhat during the course of the study.

It is difficult to draw strong conclusions from this evidence. The evidence does indicate, however, that while some advantage can be rightfully claimed for improved interface technology, the underlying service offerings may themselves lead to stable usage patterns. These experiments tested limited computer telephony services, and although the "improved" interfaces employed new technology, the new services offered small improvements over those to which they were compared. More sophisticated services can be built from the interactions between multiple functions, e.g., returning the call of a voice mail message with a single mouse click or automatically rerouting calls when the user's calendar indicates that the user is out of town. But it is much easier to measure the effects of a small, constrained change to a user-interface than it is to compare the utility of one set of features with another very different set over an extended period of use. Of the research projects described later in this chapter only MICE included an evaluation component.

Enhanced Functionality

A workstation can offer enhanced functionality by personalizing call setup for both outgoing and incoming calls. Placing a call can be facilitated by allowing the user to spell the name of the desired party with the computer keyboard, speak the name to a speech recognizer, or use another application such as a personal address book to specify the number. Or using interactive cut-and-paste or drag-and-drop paradigms, a user may be able to deposit an email message into a telephone icon to place a call to the mail sender. For incoming calls, the computer may route the call to a different phone according to a predetermined set of conditions or play a caller-specific outgoing voice mail message.

Call routing is the most sophisticated of these functions. Factors influencing how a call might be handled include the identity of the calling party, the time of day, the location and activity of the called party, and whether a receptionist is available to take an unanswered call. For example, an "important" call could invoke a distinctive ring sound or be forwarded to the nearest telephone if the recipient of the call is in another office. A call may be classified as "important" by virtue of having originated from a recently dialed phone number so as to help prevent "phone tag" in which repeated calls back and forth go unanswered. A variety of technologies can be used to determine the location of the called party or deliver notification of an incoming call while absent from the office.

Much of this potential routing information is dynamic and relies on knowledge about the user. When two parties play "phone tag" they alternate calling each other and leaving messages. If outgoing calls are placed using the same worksta-

tion that screens incoming calls, phone tag can be detected, which might initiate a more thorough search, or page, by the workstation in response to an additional incoming call. Similarly, if a meeting with someone is scheduled for tomorrow morning and that person calls today, it is likely to be relevant to the meeting and hence a more important call than otherwise, but this can be known only by accessing the called party's calendar and telephone directory to make the association.

It would be difficult to implement this degree of personalization and style of experimentation with the standard services in the telephone switch.[1] Telephone switches are designed to establish calls according to relatively static routing tables, which are modified by very limited touchtone based user interfaces. They do not have access to personal information such as a user's calendar. An entirely new dimension of flexible call establishment requires more varied per-call conditional routing factors as well as a more sophisticated user interface to manage the more complex routing rules.

Voice and Computer Access

Another motive for connecting telephones and computers is the potential for *remote voice access* to personal databases. Although a personal workstation is capable of storing many megabytes of information, its utility is limited if the data is inaccessible when the user is away from the office. Databases, such as calendars, electronic mail, and telephone numbers can be made available through interactive telephone interfaces using the techniques described in Chapter 6. Once connected to an application by voice, it is useful to be able to add to a database, e.g., to put an entry in one's calendar by speaking. Such voice interfaces may make the telephone the primary source of voice as a data type on multimedia workstations; examples will be offered later in this chapter as well as in Chapter 12.

PROJECTS IN INTEGRATED TELEPHONY

Several research projects have attempted to integrate computers and telephones in different ways. This section describes the set of telephone features of each and the system architectures upon which they were implemented. These descriptions highlight the many benefits of flexible call management by computers.

Etherphone

One of the seminal attempts to merge computers and telephones was the Etherphone project at the Xerox Palo Alto Research Center (PARC). Etherphone

[1] It should be noted that in some ways MICE, described later in this chapter, developed an architecture allowing a greater degree of user access to the configuration tables in the telephone switch. This certainly facilitates call setup services, but such tables alone cannot deal with dynamic call routing depending in real time on user input or situational information.

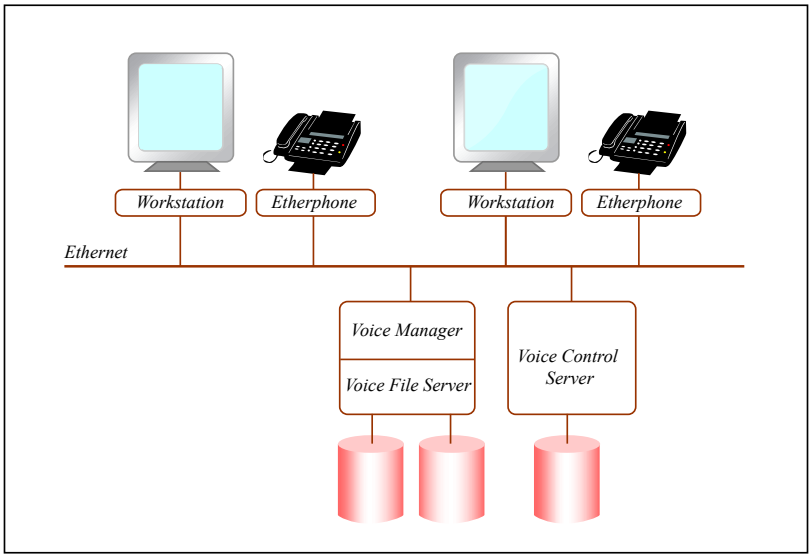

Figure 11.1. Etherphone architecture. (Figure by MIT OpenCourseWare, after Terry and Swinehart, 1988.)

included hardware to digitize voice and control analog telephone lines, both under the control of a workstation in concert with various networked servers. The project included call control applications, integration of audio with the local computing environment and its associated user interfaces, and voice-oriented applications such as audio editing, voice annotation of text, and scripted documents.[2]

The Etherphone itself was a hardware device, capable of digitizing and playing back speech over an Ethernet. Its physical components included a speaker, microphone, and a telephone handset, as well as a standard analog telephone line interface, which enabled calls to be placed over the external telephone network. Internal "calls" were transmitted over the Ethernet. During call setup, a centralized call control server negotiated with the user's workstation and then sent commands to an Etherphone agent, which dealt with the physical devices in the Etherphone and the stream of audio data packets. Voice was transmitted over the Ethernet either between Etherphones or between an Etherphone and a variety of servers as shown in Figure 11.1. The servers included a storage server to save or retrieve sound from disk, and a text-to-speech server that received text and transmitted the synthesized speech over the network.

Internal and external call processing proceeded differently within the Etherphone environment. In the case of an internal call between Etherphones, the entire conversation was transmitted over the Ethernet and local distributed pro-

[2] See previous discussion of scripted documents in Chapter 4.

cesses coordinated the setup and tear down of the call. For external calls, the Etherphone provided ring detection for incoming calls and touchtone generation to place a call, but switching occurred outside Etherphone in the ordinary telephone network.

For basic telephony, Etherphone provided several methods of dialing a number including a personal directory and dialing by name. A log was kept of all call activity. Tools, primarily text-based (see Figure 11.2), provided user interfaces to the dialing functions. For incoming calls, each Etherphone user had a distinctive ring "tune" to distinguish whose phone was ringing when within earshot of several offices. If the caller was also an Etherphone user, his or her ring tune alternated with that of the called party, giving an audible form of calling party identification. A telephone icon on the screen also indicated the incoming call and visually displayed the caller's identity.

With local call routing and voice distribution over the Ethernet, a number of new services were possible. The tight binding between a name and a telephone number could be broken; while the called party was logged in on any workstation, Etherphone calls could be forwarded to the nearest Etherphone. While in another office, an Etherphone user could enable "call visiting" from any workstation, causing one's calls to forward to that office. Because incoming calls were indicated with the distinctive ring tune, the host and visitor knew who should answer the telephone. A meeting service allowed multiple listeners to tune in to a meeting

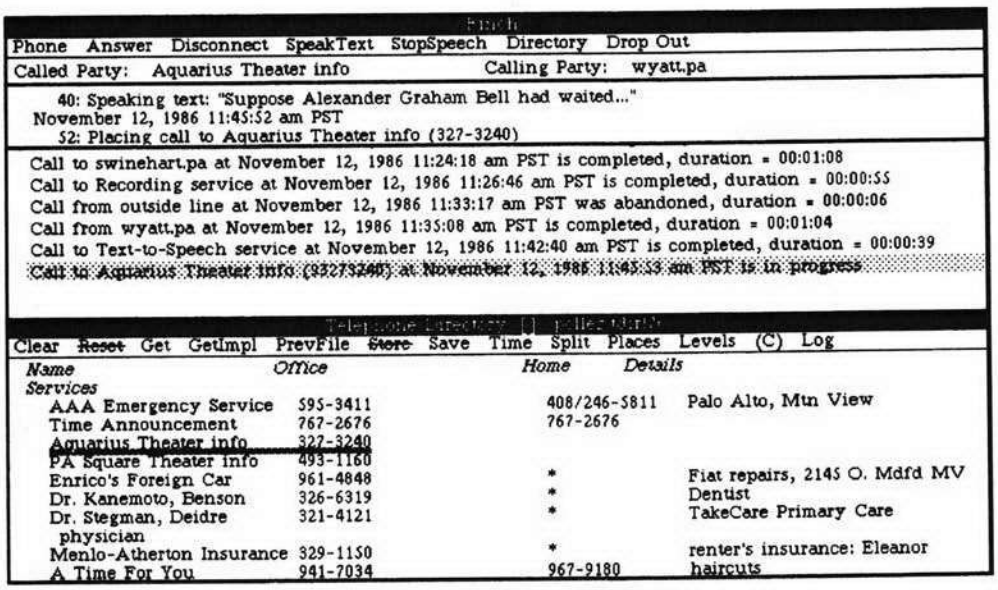

Figure 11.2. Etherphone telephone tools. (From Zellweger, Terry and Swinehart, "An Overview of the Etherphone System and its Applications." Reprinted with permission from *Proceedings of 2nd IEEE Conference on Computer Workstations* ©1988, IEEE.)

or lecture from their separate offices; remote participants could speak to the meeting as well as listen in. The meeting service was implemented by multicasting the voice packets from the meeting room Etherphone to those of the parties listening in.[3]

Etherphone implemented fully-distributed call control by allowing workstations to place calls through requests to the connection server, i.e., a "software PBX." Both voice and call control data passed through the local area computer network. Calls originating outside the Etherphone service area were also handled in a distributed manner as each workstation's Etherphone interfaced directly to the outside analog lines.

Through its layered architecture, Etherphone brought a greater degree of sophistication to call management. The flexibility of its hardware and network implementation enabled broadened concepts of a telephone "connection." As a highly integrated environment, Etherphone made major contributions for voice applications development in the context of a larger operating system; such styles of integration are the focus of Chapter 12. It was known for its variety of applications for voice in documents as discussed in Chapter 4. Finally, Etherphone helped extend the concept of *services* by partially bridging the gap between the services offered by the telephone industry such as call forwarding and those offered by the computer industry, such as file storage.

MICE

MICE (Modular Integrated Communications Environment) at Bellcore (Bell Communications Research) was built as a testbed for prototyping enhanced services and evaluating their use over time. The subjects of these evaluations were local laboratory members, who used MICE on a daily basis from 1985 to 1988. Although many of the trial services implemented under MICE were fairly simple, MICE provided a foundation for forging closer links between the workstations and telephones in each person's office. The trial services included "memory dialing," a mnemonic form of speed dialing, and MICEmail, an integrated voice and text message system. MICEmail provided both local and remote (using speech synthesis) access to both types of messages.

MICE explored several issues from the point of view of a local telephone operating company by defining the services and resources to be accessed over a future telephone network. MICE placed emphasis on *personalization;* messages were sent to users instead of terminals with the network turning the name into an address and determining a suitable delivery mode. These services were designed to be *customizable;* the user-configured services by changing a profile table within the MICE system. Finally, MICE worked with *integration* of voice and text message types.

[3]Broadcast sends the same packet to every host on the network. Multicast also sends a single packet, but only selected hosts read it.

The MICE architecture was based on a centralized server, which provided basic telephony functions as well as voice storage, speech recognition, and text-to-speech synthesis. Telephony functions were provided by a small PBX closely controlled by a workstation, which acted as a server to other computers on the local area network. A central control process on the server controlled the telephone switch by executing a finite state machine defined by a call control table, which was accessible to other workstations on the network. The call control tables could be programmed to implement the new server by means of a graphical "service editor," but making changes to the tables was a complicated procedure. Some work was done to explore screen-based interfaces to such simple service definitions such as time-of-day dependent call forwarding.

MICE was considered to be a prototype open-architecture central office telephone switch in which switching is performed entirely within the switch but outside entities can program how switching occurs. Although its primary purpose was to provide a flexible service development environment for its creators, MICE

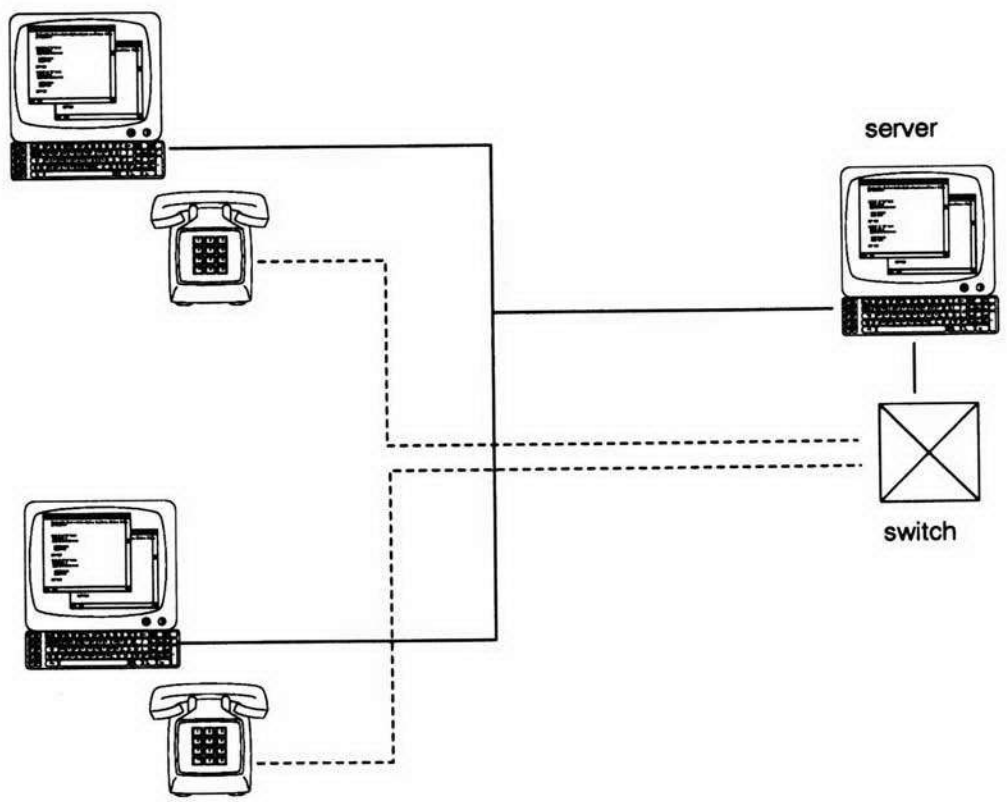

Figure 11.3. MICE architecture. Desktop computers interface to telephones via a server computer controlling a telephone switch.

also allowed computer access to the call control tables that described how calls were handled. Client workstations did not handle each call as it came in; rather, they could update the tables which the server used to route subsequent calls.

Being centralized, MICE transmitted audio to users' offices for services such as voice mail over the telephone network. Even when using a screen-based interface to access voice messages, a user was required to pick up the telephone to hear a message. Using telephone wiring to transmit audio from a specialized server was advantageous at a time when each workstation could not play audio locally. However, the inconvenience limited voice to a few applications instead of allowing it to become a widely available data type to multiple applications.

BerBell

BerBell,[4] also from Bellcore, was used for telephone and voice services for a real user population, and it offered increased flexibility for the creation of new services under control of the users' workstations. The Berbell project was initiated in 1985, became operable in 1986, and continues in use today. Some of the more well-known services provided by Berbell include speech synthesis for a national weather forecast, a music synthesis demonstration, and local voice paging using speech synthesis over a public address system.

Berbell utilized three servers: one for the telephone switch, one for speech synthesis, and another for digitization and playback of audio. The system included a number of basic services and features that could be invoked by the user with touch tones; in this sense, it was not very different from many PBXs. But since the Berbell servers were also on the local computer network, the same features could be activated from the user's workstation. Additionally, a programmatic interface allowed users to program new services. Over its lifetime, Berbell evolved in the direction of providing complete control over telephone calls at the workstation; in this sense, it allowed much more dynamic call control than MICE.

Personal eXchange

The Personal eXchange (PX) project at the Computer Research Laboratory of Bell Northern Research took a distributed approach to the integration of voice and telephony with computer workstations [Kamel, Emami and Eckert 1990, Bowles *et al.* 1991]. PX also employed a central digital switching facility (a Northern Telecom Norstar PBX); however, this was an "open" PBX, and all routing commands were delivered by computers and telephones distributed around an office. As shown in Figure 11.4, located on each desk were the following.

- A telephone used for its microphone, speaker, and to serve as backup equipment if the computer failed.

[4]Named after the login ID of its creator, Brian E. Redman.

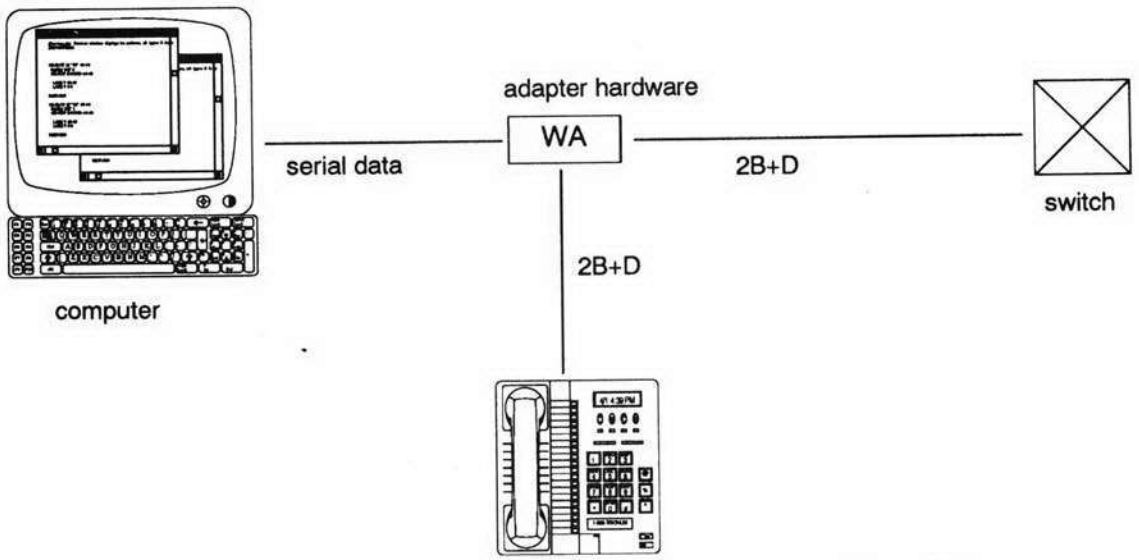

Figure 11.4. Architecture of the PX project. A workstation adapter, WA, allows a computer to interface to both the telephone instrument and the telephone switch by converting digital signalling protocols.

- A "workstation adapter" that brought both speech and signaling information to the computer via a digital telephone line.
- A computer, that controlled the telephone set in addition to call processing.

The wiring of the digital telephone system was used for both voice transmission and exchange of messages between call management applications and the telephone switch. Protocols similar to ISDN allowed simultaneous transmission of voice and signaling messages. When a call arrived, the switch broadcast a call setup message to all workstations with access to the dialed number. Multiple recipients could respond; for example, a screen-based telephone (such as that shown in Figure 11.5) or the telephone set next to the workstation could try to claim the call. Whichever recipient responded to the call sent an acceptance message to the switch and a voice connection was established.

A wide range of applications were implemented around this architecture; the telephone dialing interface of Figure 11.5 is but one example. In another PX application, a conversational answering machine took an interactive message by playing prompts and recording answers, and a graphical user interface allowed message retrieval (see Figure 11.6). Towards the end of the project, speech recognition and speaker verification were being incorporated into the workstation as part of the set of local services available to applications. Although much of PX focused on providing mechanisms for distributed call management, it was also concerned with the larger issues associated with the management of voice in the workstation (such as the voice editor described in Chapter 4) along with software toolkits to support audio and telephony in applications.

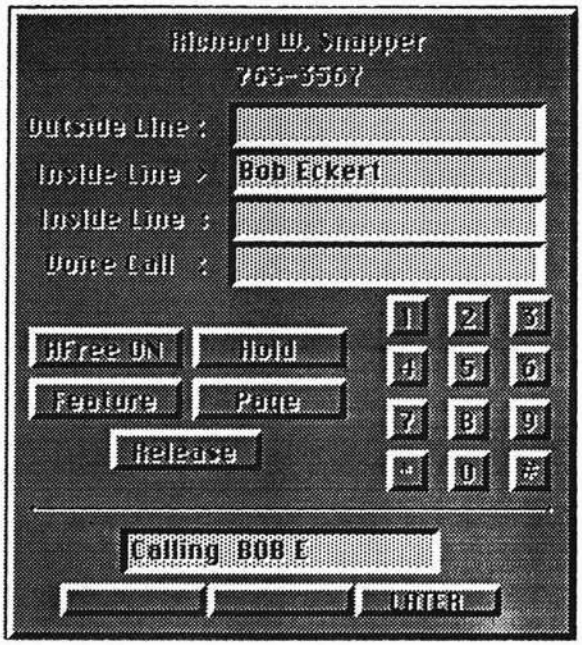

Figure 11.5. A PX screen-based telephone application. (Printed by permission of BNR.)

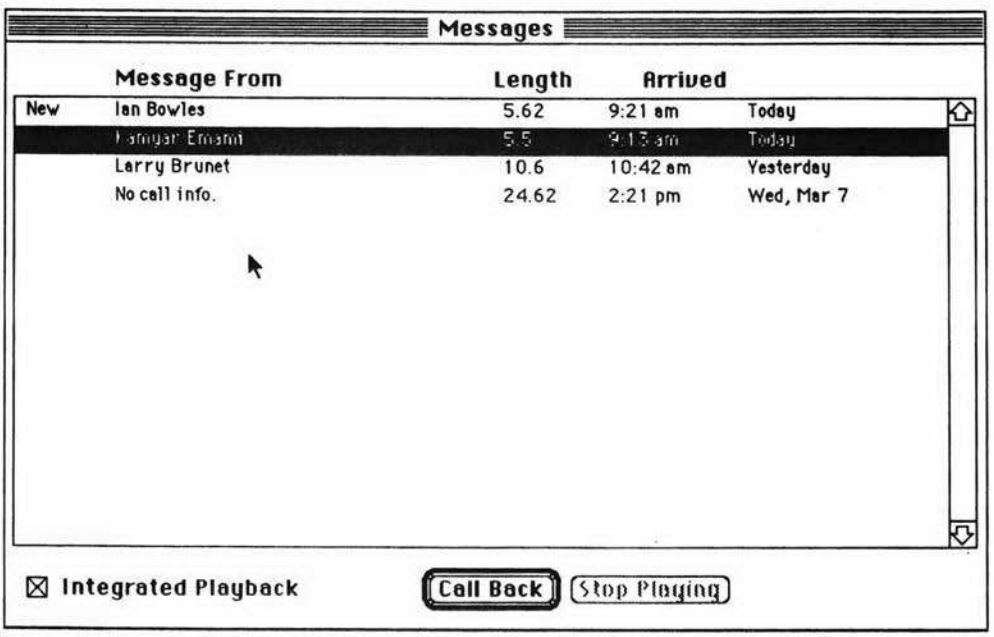

Figure 11.6. The display of a PX answering machine program. (Printed by permission of BNR.)

Except for the PBX, PX was a completely distributed environment. The connection management decisions were made on individual workstations. Each workstation implemented its own voice services and managed recorded voice locally. In addition to the speaker built into the computer, the local telephone set could be used as a speaker and microphone by the computer.

Phonetool

Phonetool is a telephone dialing tool that allows users to place calls from their workstation; it was developed by Stephen Casner at the University of Southern California's Information Sciences Institute. Phonetool runs under the Sunview window system and uses a centralized telephone server to place calls. Phonetool provides a pop-up speed-dial menu (see Figure 11.7), can redial the most recently dialed number (which is displayed on the telephone icon), accepts numerical input from the keyboard, and can dial a number displayed in another window using mouse-based cut-and-paste interaction.

Phonetool places calls using a centralized server; a server is required because of hardware interface limitations for access to the telephone switch (a Rolm PBX). The server can place outgoing calls but it is not provisioned to receive them. Software was written to allow a personal computer equipped with a special PBX interface card and an Ethernet controller to function as a centralized server, i.e.,

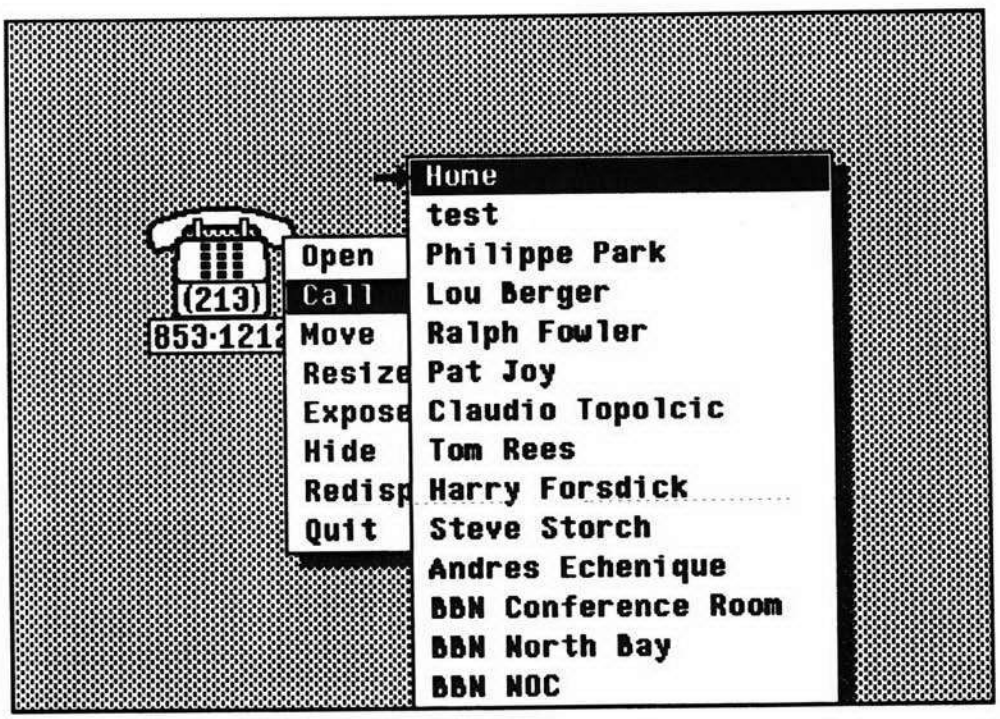

Figure 11.7. Phonetool's speed dialing menu.

by dialing on behalf of the numerous workstations (see Figure 11.8). In order to dial, the user's workstation sends a message over the Ethernet to the server specifying the number to dial. The server then dials the call on its own outgoing phone line. At this point, the server puts the outgoing call on hold and dials the user's intercom extension. Because the intercom is used, this call can go through automatically, putting the called party's telephone into speakerphone mode. At this

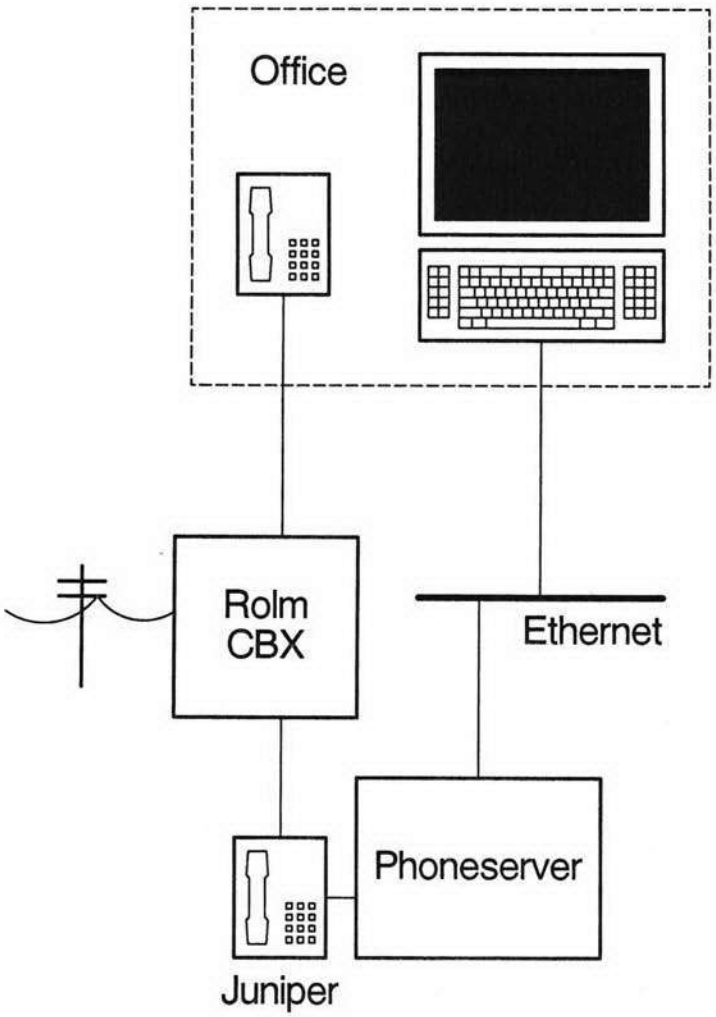

Figure 11.8. The I.S.I. phoneserver utilizes a personal computer with a specialized interface card to control the PBX. Workstations make requests of this server over a local area network. (From Schmandt and Casner, "Phonetool: Integrating Telephones and Workstations." Reprinted with permission from *IEEE Global Telecommunications Conference* © 1989, IEEE.)

point, the server puts the intercom call and the outgoing call together as a conference call and proceeds to drop out of the conversation. The server is now ready to service another request.

Phonetool was the starting point for Xphone described later in this chapter. Separating call control functions from the user interface allowed Xphone to be configured for both centralized and distributed architectures in a number of implementations.

ARCHITECTURES

Several different system architectures can provide telephone functionality to workstations. It is useful to differentiate the *hardware* interface, which physically connects the computer to the telephone switch, from the *functional* interface, whereby applications on the desktop implement telephony services. A **distributed** hardware architecture requires each computer to have a hardware interface for communication with the switch. A **centralized** architecture uses a single link to the switch from one workstation; other workstations access this hardware over the network by communicating with a server process on the specially equipped workstation. Even when a hardware interface is found on each workstation, a local server may be used to allow multiple client applications to share hardware access. All these architectures could offer identical functional interfaces to clients for call control; what differs is the mechanism whereby client requests are communicated to the switch.

Distributed Architectures

Enhanced telephone sets on every desk constitute the most direct implementation of a fully distributed telephone hardware architecture (see Figure 11.9). Advanced telephones may include improved displays, keyboards, touch screens, or speech recognition. These telephones provide superior user interfaces to traditional telephone functionality provided by the switch, and may implement new services, such as electronic mail storage within the telephone. This is not a new concept; in the early 1980s a number of "teleterminals" appeared providing small CRT displays and keyboards. More recent teleterminals use LCD or plasma displays with features such as programmable function keys, stylus input for hand writing, and local voice and text storage for messages. The most recent generation of enhanced telephones is ISDN-based; the ease of data communication between telephones allows for a range of new message services such as digital facsimile and screen-based remote data access.

This approach sidesteps the problem of computer integration, making the telephone powerful enough to obviate the need for the computer. But such devices have not proven popular beyond such consumer products as the combined telephone and answering machine. When made powerful enough to provide support for sophisticated teleservices, such stand-alone telephones become too expensive. Business users already use personal computers or workstations at their desks,

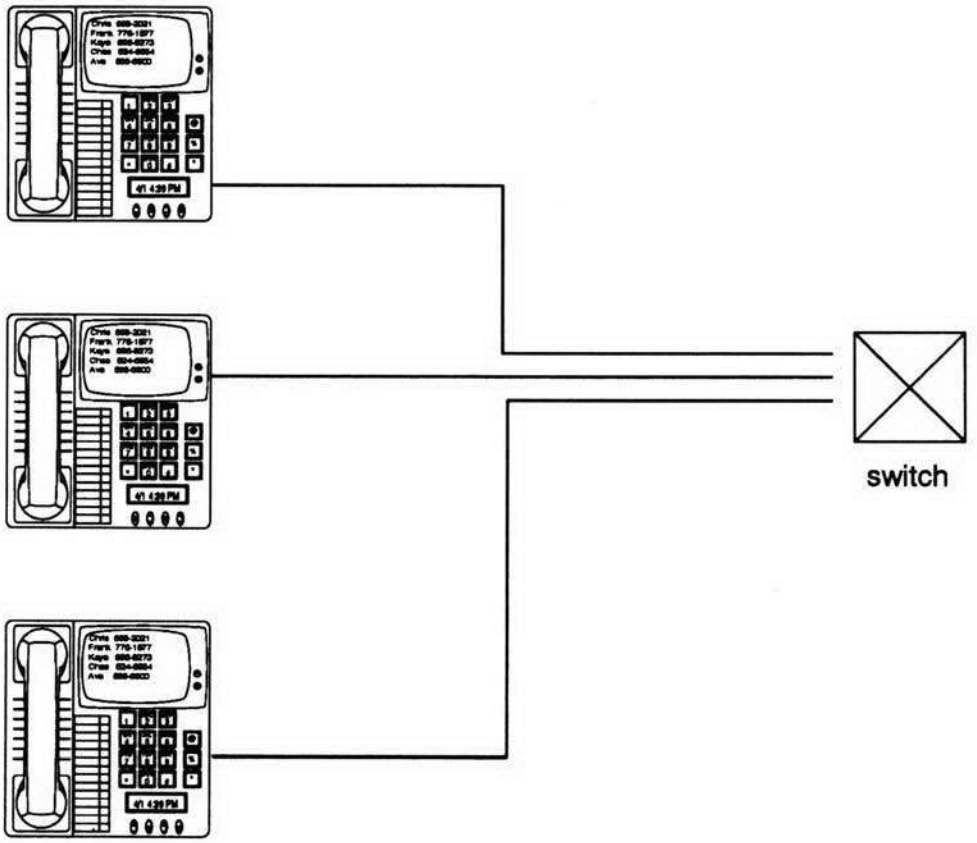

Figure 11.9. A distributed architecture of enhanced telephones.

and the general-purpose computer provides for more efficient resource allocation. Although much can be done to improve the design of telephones, they are not destined to push computers off the desktop. However, in environments where computers are not found, the introduction of ISDN may create a niche market for enhanced telephones.

The teleterminal provides a completely distributed architecture with one telephone per office. A completely distributed architecture can also be used for workstation-based telephony. For example, an inexpensive modem wired in parallel with a telephone can be used with autodialing software; the computer places the call and the user picks up the handset if the call is answered. Similarly, new consumer products plug into an analog phone line and decode calling-line ID information; when this information is available, the calling number is transmitted over a serial line to a host computer. Such approaches are practical on analog lines because modems are inexpensive and computer serial ports ubiquitous, but they provide only limited signaling functionality. For the computer to talk or listen to the voice circuit during a call, a higher degree of integration is necessary.

One means of providing both call control and telephone audio to a personal computer is a specialized telephone interface board. Such a board includes the required electrical interface to allow it to be connected to the telephone network, touchtone generator and decoder integrated circuits, a codec, and perhaps a digital signal processor or an integrated circuit for speech compression. Telephone interface boards are currently available for as little as a few hundred dollars.

Many workstations and personal computers now support audio digitization and playback. Telephone functionality can be added using simple external electronics. An external unit provides a telephone jack, a serial interface for call control, and analog audio input and output to be connected to the workstation. The computer sends commands and receives notification of events such as an incoming call over the serial interface. When the phone line is off hook, the computer's audio input and output are connected to the line. Touch tone dialing can be accomplished by playing prerecorded touch tone sounds once the telephone has been taken off hook. Tone decoding can be performed in the workstation by software (if the workstation is powerful enough) or by an integrated circuit in the external hardware.

ISDN is very attractive for workstation integration of telephony because it provides finer-grained call control due to its richer signaling protocol. The appropriate electrical hardware interfaces are beginning to appear as standard equipment on workstations, with ISDN call control protocols implemented in software. Once a telephone connection is established, the voice channel is already digital so further hardware is not required for voice interaction. Current advanced workstations can easily run Layer 3 ISDN protocols in software under multitasking operating systems. But less powerful computers or operating systems require separate microprocessor-based peripheral hardware to handle the real-time requirements of ISDN. Such peripherals are currently selling in the $800 range, which may represent a significant portion of the cost of the computer itself.

Centralized Architectures

In the distributed architecture just described, telephone interface hardware is located on every computer. Such an arrangement may be impractical or inadequate for some applications, however. For either proprietary or ISDN digital switches, telephone interface hardware may be prohibitively expensive. Computer reliability is also an issue; when the workstation is down, does the telephone still work? Although it is easy to configure an analog line control card with a second telephone jack to let an adjunct telephone function when power is turned off, this is much more difficult under ISDN signaling. Some PBXs offer sophisticated computer interfaces but are equipped to connect only a single computer (usually through a serial interface); this precludes direct switch access from each workstation.

For any of these reasons, a centralized hardware architecture may be preferred or required. Figure 11.11 illustrates an example of this approach; using a telephone server provides distributed access to a network of computers. One computer, the server, includes the required hardware and software to communicate with the telephone switch on behalf of its clients. Clients are distributed over a

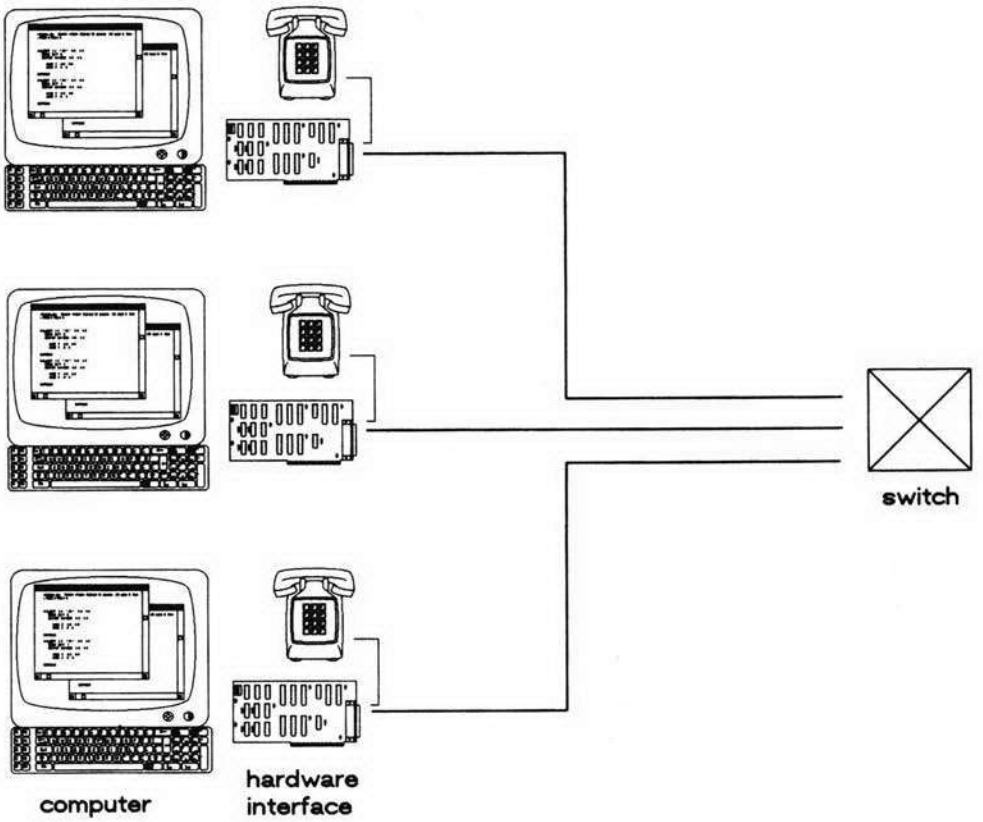

Figure 11.10. A distributed architecture with a telephone peripheral on each workstation.

local area network, sending requests and receiving events (such as notification of an incoming call) from the server. In this centralized architecture, application requests are translated into messages for the server and sent over the computer network; the server, in turn, translates these into commands to its local hardware and thence onto the telephone network. The centralized approach requires that one computer interact with many telephone lines so that a single server can act on behalf of a number of clients. This can be accomplished in several ways: each line can be connected to the server or the server can communicate directly with the telephone switch.

With analog telephony, each line has its own independent circuit for signaling and voice. A multiline analog telephone or key set has a pair of line wires for each line and selects between them with buttons. Similarly, interfacing a computer to multiple analog lines requires the use of one telephone line interface board per line. A single board can be made to handle multiple lines by replicating the interface circuits, which reduces the per-line expense; consequently, such boards are often used in voice mail systems, serving four to sixteen lines per board.

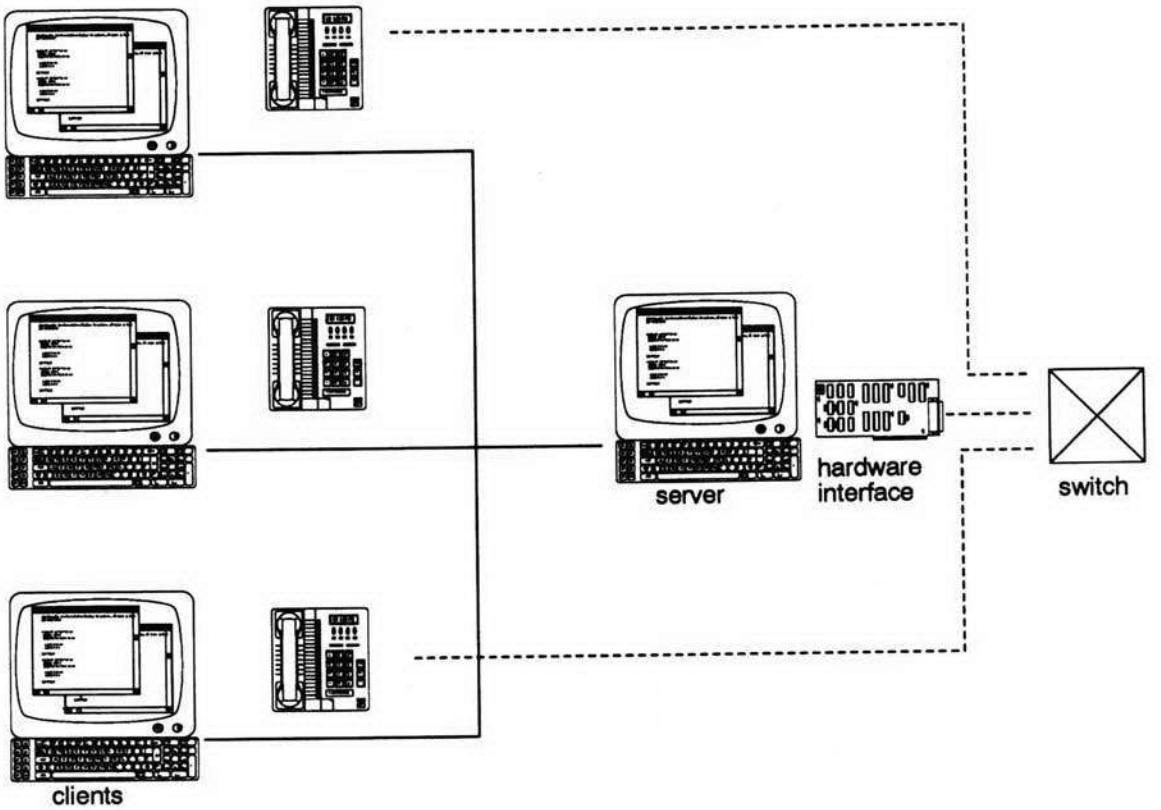

Figure 11.11. A server allows centralization of some resources with a network interface to each workstation.

An alternative arrangement is for the server to communicate with the switch directly through a custom interface. Some experimental systems, such as MICE, use small PBXs designed to be controlled by a host computer. Larger PBXs are less flexible but may still provide a simple computer interface. For example, some switches support a message set called SMDI (Simple Message Desk Interface), which carries selected signaling information from the switch over a serial line. Interfaces such as SMDI are often used by voice mail systems in the following fashion: When a call gets transferred to voice mail, the switch identifies the originally dialed number and a trunk line switches onto that which the call has been transferred so that the voice mail system can record a message into the proper mailbox. The signaling interface is bidirectional so the voice mail system can control the telephone's message waiting light via the switch. Such a PBX interface can extend call control to a single workstation, which communicates with the PBX and acts as a server to other workstations. SCAI (Switch-Computer Application Interface) is a proposed standard for external switch control. It is currently under study by the T1S1 committee of ANSI (the American National Standards Institute).

For digital telephone systems, particularly ISDN, multiple telephone numbers can appear on a single phone line if the switch is configured to support key set style telephones.[5] The sole D channel for the ISDN line carries signaling information in both directions for all numbers appearing on the line. So if in addition to each user's personal telephone line another line is configured with appearances of each user's number, it can be used by a server to control calls on those numbers. The D channel on the server's line can be used to allow a server to answer, transfer, or place calls by proxy at the request of client applications running on other workstations not equipped with ISDN interfaces. If the server ceases to function, the user's office telephones are unaffected.

Comparison of Architectures

The ISI Phoneserver described earlier is an example of the server approach with centralized hardware. The choice of architecture was dictated by the PBX interface board; this board was expensive and incompatible with the users' workstations in their offices. Both MICE and Berbell used centralized telephone switching because the switching hardware supported a single serial interface. MICE stored call control tables in the server; instead of routing each call, clients programmed the call control tables and the server routed calls on its own. By contrast, Berbell allowed clients to dynamically route calls. MICE and Berbell also used centralized speech resources such as codecs and speech synthesizers. Although this provided adequate support for services to interact with remote callers, a user in an office also had to pick up the telephone to access these devices, e.g., to hear a voice message.

An ISDN-based telephone server similar to Berbell yet lacking built-in voice storage has been built at the MIT Media Lab. This server, used by applications described later in this chapter, allows access to a cluster of phone numbers using normal D channel signaling, obviating special hardware access to the telephone switch. In this environment only call control is centralized; audio services are provided locally at each workstation. Several servers provide specialized services such as a voice mail server to answer calls using the telephone server's B channel interface.

Both PX and Etherphone used fully-distributed architectures that took advantage of additional centralized services. In PX the telephone switch broadcasted information about every call, and the appropriate workstation requested that the switch complete the voice circuit. Although the switch was a central resource, it was designed to support fully-distributed call management, and most importantly the digitized voice was routed directly to the workstation. Similarly, in the Etherphone project, a central conversation manager connected digital audio streams from separate workstations for internal calls, whereas each workstation had complete local control to place outgoing calls and receive incoming calls from

[5] Unfortunately, these features are not published as any ISDN standard, so for now they remain vendor specific.

the public telephone network. PX stored sound locally on the workstation, while Etherphone digitized it and then stored it via a centralized voice storage server utilizing the voice ropes database described in Chapter 4. PX workstations shared a centralized speaker verification service, and Etherphone allowed workstations to have access to a central text-to-speech synthesis service.

From the point of view of hardware and system maintenance, the centralized and distributed architectures differ radically. These hardware differences can be made largely transparent to applications by means of a software library that shelters an application from details of communicating with the remote server. If implemented in such a network-transparent manner, the user is not concerned with whether the server process is on a local or remote workstation.

The server approach is attractive for device sharing and resource arbitration even if hardware devices and application processes are located on the same workstation. Device sharing promotes the development of environments with multiple small applications rather than a single monolithic application: a server allows each application to operate without knowledge of other applications that may require sporadic access to limited resources. Multiple applications may receive simultaneous notification of an incoming call; one application might alert the user using speech synthesis to name the caller or displaying the caller's name in a window on the workstation. Concurrently, an answering machine (voice mail) application waits a specified amount of time after the call is announced and picks up if the called party does not answer. Another application may keep a log of all calls, incoming or outgoing. A server architecture promotes such sharing of hardware resources, whether local or remote.

The differences between centralized and distributed architectures is more significant for voice storage, transmission, and presentation to the user in an office. If the telephone line terminates in the workstation, then when the workstation answers a call the voice circuit terminates locally. Under this architecture, an application might allow an incoming voice message to be monitored on the local workstation speaker for call screening and also permit the called party to pick up the phone in the middle of taking a message. Locally stored voice messages[6] can be played through the speaker without needing to re-establish an audio link to the server.

When the telephone line terminates in a remote location, monitoring message recording is more difficult and call pickup may not be possible. Losing the ability to screen calls is a common complaint among users when voice mail replaces answering machines. If voice is stored on a central resource, then it can be transmitted over the computer network (as was done in Etherphone) in order to be played. MICE used the existing telephone wiring to playback voice messages by first placing a local call from the central storage manager to the telephone in the office. This required the user to answer the phone to hear the message, making

[6]The actual disk drive on which the voice is stored may be in another location remotely mounted over the local area network. But this is transparent to the application and the sound appears as a local file system.

message playback much less spontaneous. Although we are used to the telephone as the means to retrieve voice mail, when stored voice permeates many desktop applications such as described in Chapter 12, audio playback through local speakers is much more desirable at least for use in a private office.

CASE STUDIES

This section describes several sample projects from the MIT Media Lab. These case studies are intended as examples of the services and user interfaces that can result from the integration of telephony applications with computer workstations. Some of the case study examples contain similarities to the work on integrating computers and telephones discussed earlier in this chapter as many of the ideas have been explored across multiple research labs.

Phone Slave

Phone Slave was an early (1983) integrated telephone management system developed by the MIT Architecture Machine Group (predecessor to the Media Lab) [Schmandt and Arons 1985, Schmandt and Arons 1984]. Phone Slave provided a variety of telephony functions including message taking, personal outgoing message delivery, a telephone number and address database, speed dialing by name, and remote access to both voice and text mail. With the breadth of functionality it supported, Phone Slave attempted to provide a complete interface to telephony functions from a general-purpose computer.

Phone Slave included both graphical and telephone-based user interfaces. It did not run in concert with any window system as window systems were not in general use at that time. Phone Slave was a monolithic process; in subsequent work at the Media Lab, many of Phone Slave's functions were implemented as separate applications (two examples, Xphone and Xrolo, are described later in this chapter). In the absence of a window system, Phone Slave shared the single graphics screen between three different display views: one for message retrieval, one for dialing on a telephone keypad, and the last for the name, address, and number database.

Phone Slave adopted a conversational approach to taking a telephone message. It asked a series of questions and recorded each response in a separate file. During recording, an adaptive pause detection algorithm determined when the caller was finished; the algorithm was adaptive with respect to the background noise level as well as the speech rate of the caller. The sequence of Phone Slave's questions was as follows.

- "Hello, Chris's telephone, who's calling please?"
- "What's this in reference to?"
- "I'm sorry, he's not available now, but he left this message . . ." followed by the owner's outgoing message and then "At what number can he reach you?"
- "When will you be there?"

- "Can I take a longer message for you?"
- "Thanks, I'll give him your message, goodbye."

Some naive subjects did not realize that they were speaking to a computer (note the wording of the initial greeting). Of course, there were instances when the conversational model would break down, which most commonly occurred when the caller responded simply with "yes" to the last question. The caller proceeded to wait for an anticipated prompt to leave a message and was surprised by the closing sign-off announcement. Each recorded comment with the exception of the last had a maximum anticipated response duration; once this was exceeded, Phone Slave would interrupt, saying "Excuse me, I'm just an answering machine, and it would be helpful if you could answer my questions directly."

Phone Slave's conversational format served several purposes. The first was to make it easier for the caller to leave a complete message. At the time when Phone Slave was implemented, answering machines were uncommon and seldom found in business situations so not all callers were accustomed to them. The second purpose was to provide segmentation of the messages for the sake of the recipient, Phone Slave's owner. Finally, during the caller's response to the first question (*"Who's calling please?"*), the telephone line audio output was routed to a speaker-dependent speech recognizer that attempted to identify the caller. The caller had to speak a consistent greeting to be recognized as the recognizer merely detected the familiar pattern of words.

A caller might be identified by Phone Slave, either by speech recognition or by entry of a touchtone password at any point in the conversation because speech recognition over long-distance telephone lines tended to be unreliable. If recognized, the caller was greeted by name, informed if Phone Slave's owner had listened to any recent messages from the caller, and possibly received a personalized outgoing message. After this, Phone Slave would say "If you'd like to leave another message, I'll record it now. Otherwise, just hang up and I'll tell him you called again."

When the owner called in and logged in, he could take advantage of a number of Phone Slave options including hearing messages, deleting messages, changing the outgoing message, and recording personal messages for known callers. Commands (e.g., Play, Delete) were entered either by speech recognition or touch tones. Because the incoming messages were segmented by the conversational recording method, several novel review options were available. For example, messages could be scanned quickly by asking "Who left messages?"; the system responded by playing back the initial recorded segment for each message in which the callers would have identified themselves.

Unlike conventional voice mail, Phone Slave integrated voice and text (electronic mail) messages. Text messages were presented by speech synthesis using the user interface described in detail in Chapter 6. Synthesized and recorded messages were intermixed, as Phone Slave sorted messages according to the sender (rather than the conventional time-sequential order) to facilitate complete responses.

In addition to taking messages, delivering personal messages, and providing mixed-media message retrieval over the telephone, Phone Slave provided screen-

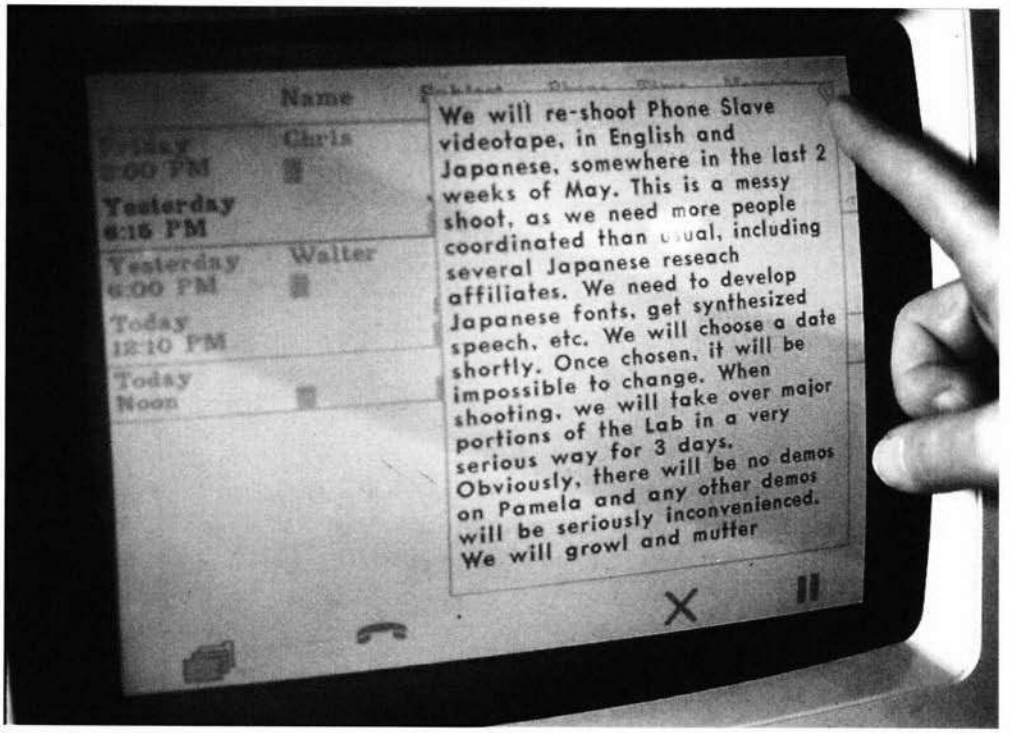

Figure 11.12. Phone Slave used a pop-up window to display a text message. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the *1984 SID International Symposium Digest of Technical Papers, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146–149.*

based message access as well as other telephony functions using a color display and a touch-sensitive screen. Phone Slave implemented a limited set of window system functions, sharing the screen between a number of "views" and using pop-up windows for email display (see Figure 11.12). The message access view, for example, displayed each message with text fields to indicate the caller (if known), date and time, and subject field (for a text message). Each recorded sound segment appeared as a bar (see Figure 11.13), whose length indicated the duration of sound. Each message was displayed as a row of sounds; touching the left edge of the row would play each message segment in sequence. The bar changed color during playback from left to right in synchronization.[7] Touching a text message would pop up a text display window that could page through the message (see Figure 11.12).

[7]This sound bar was a direct predecessor of the SoundViewer widget for the X window system described in Chapter 12.

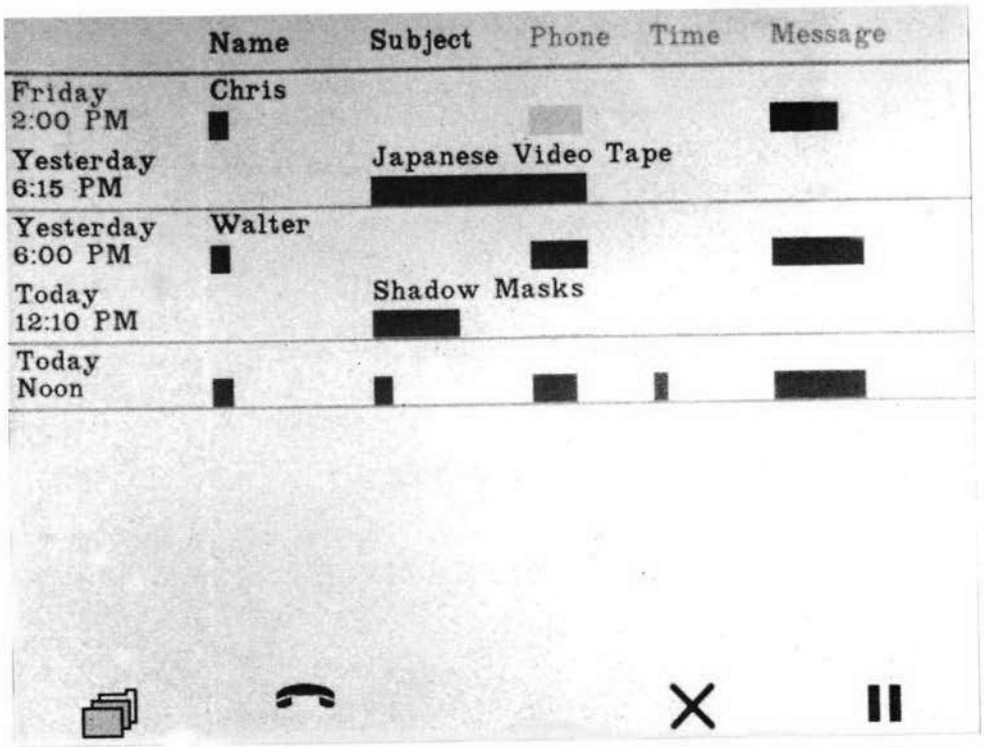

Figure 11.13. Phone Slave's message access view. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the *1984 SID International Symposium Digest of Technical Papers, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146–149.*

Phone Slave could place calls in several ways. The simplest method involved bringing up the telephone dialing view by pressing the telephone icon at the bottom of any other view (see Figure 11.14). The dialing view was modeled after a telephone keypad with an area for dialed-digit display. As with Phonetool discussed earlier in this chapter and Xphone introduced in the next section, Phone Slave accumulated the appropriate set of digits for a complete telephone number before commencing to dial, which allowed the user to delete digits without needing to hang up and restart from the beginning. While a call was in progress, the handset in the dialing view was raised to indicate the off-hook state; to hang up, the user would touch the handset, which would drop it back in place.

Phone Slave also included a simple name, address, and telephone directory, which was displayed in a view that listed each entry as an alphabetized card. This database was created in a text editor; Phone Slave allowed read-only access to it. In addition to the expected fields on the card, such as address and telephone number, any personalized outgoing messages were indicated by the presence of a

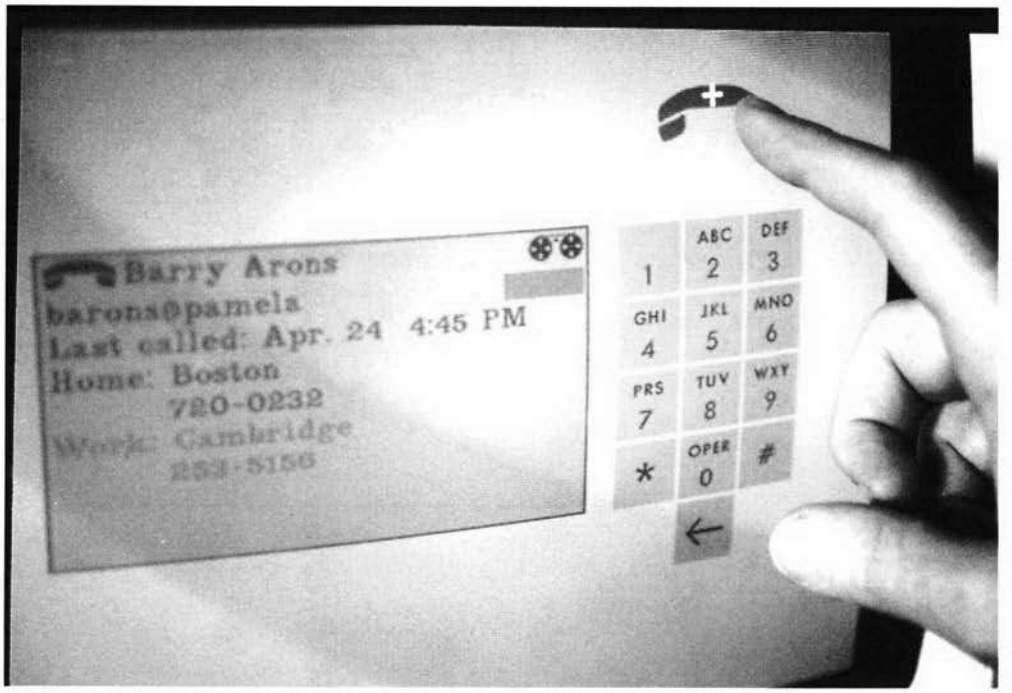

Figure 11.14. Phone Slave's telephone dialing view. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the *1984 SID International Symposium Digest of Technical Papers, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146–149.*

sound bar on the card. The owner could review the outgoing message by touching the bar or create new messages by touching on the "reel of audio tape" icon. Depending on the time of day, either the home or work number would be highlighted in red. Touching the telephone icon on the card would switch to the telephone keypad view and automatically dial the number.

A special card was dedicated to choosing the generic outgoing message for an unknown caller (see Figure 11.16). The owner could select from a number of previously recorded messages suitable to particular occasions (out to lunch, out of town, etc.) or record a new temporary message. This option was included as part of the overall project goal of maximizing the information content of a transaction with an answering machine; as much information as possible was provided to both known and unknown callers.

Phone Slave was a direct predecessor of several related projects that followed in the next seven years. The answering machine portion was included in the Conversational Desktop project described in Chapters 9 and 12. Screen-based voice mail has recently been resurrected in a multiprocess version running under the X Window system. Separate processes manage message taking, the graphical

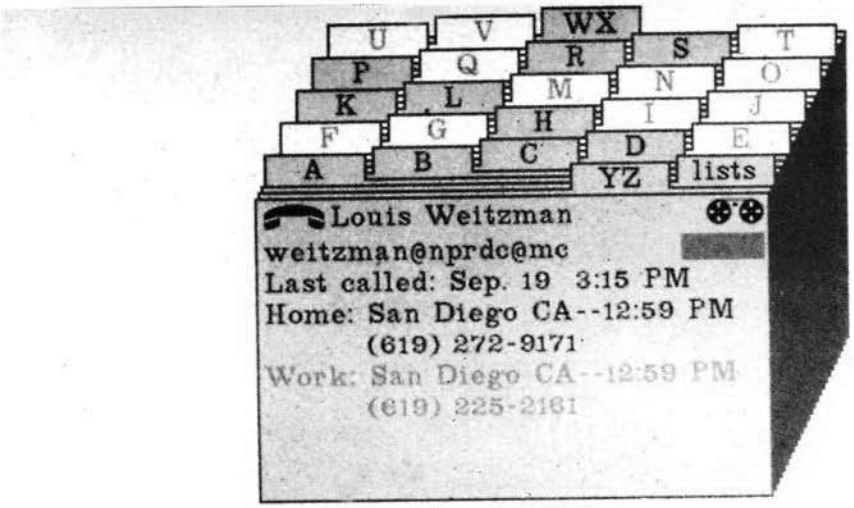

Figure 11.15. Phone Slave's name, address, and telephone number directory view. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the *1984 SID International Symposium Digest of Technical Papers, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146–149.*

user interface, and remote telephone access to messages; this is described in more detail in Chapter 12. Also, Xphone, a telephone dialer, and Xrolo, a telephone directory application, provide the functionality of two of Phone Slave's views.

Xphone and Xrolo

Xphone is a telephone dialing tool based in large part on the Phonetool dialing tool developed by Stephen Casner and described earlier in this chapter. Xphone runs under the X Window system and offers increased functionality over the original Phonetool [Schmandt and Casner 1989]. The basic idea remains the same: a powerful telephone dialer that occupies minimal screen real estate.

Xphone is displayed as a small window showing the most recently dialed telephone number (see Figure 11.17). There are a number of different methods of

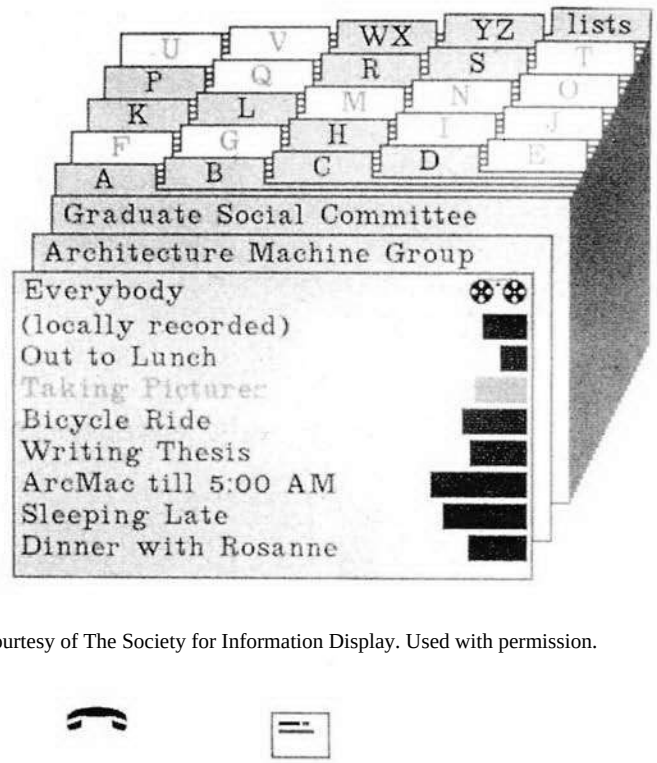

Figure 11.16. The outgoing message was selected using a special card. From Schmandt and Arons, "Phone Slave: A Graphical Telecommunications System." Reprinted with permission from the *1984 SID International Symposium Digest of Technical Papers, edited by Jay Morreale. Vol. 25, New York: Palisades Institute for Research Services, Inc. June, 1984. pp. 146–149.*

dialing a number with Xphone. The most recently dialed number can be redialed by clicking the left mouse button on the window. A number can be selected with the mouse from a different text window; when it is pasted into Xphone, the number is dialed. Alternatively, the user can type a number into the tool using either the number keys or numeric keypad on the computer keyboard. The keyboard-entry method is not as comfortable for dialing as the telephone keypad because the keys are arranged in a different order, but once the number has been entered, it can be dialed again much more quickly either via redial or the phone log described shortly.

Holding down the right mouse button pops up the speed-dial menu (see Figure 11.18). This menu displays the names found in a simple text file. A submenu reveals a log of recently dialed numbers (see Figure 11.19). Note that an entry in the log may appear as either a name or a number depending on how it was dialed.

Figure 11.17. The Xphone window displays the most recently dialed telephone number.

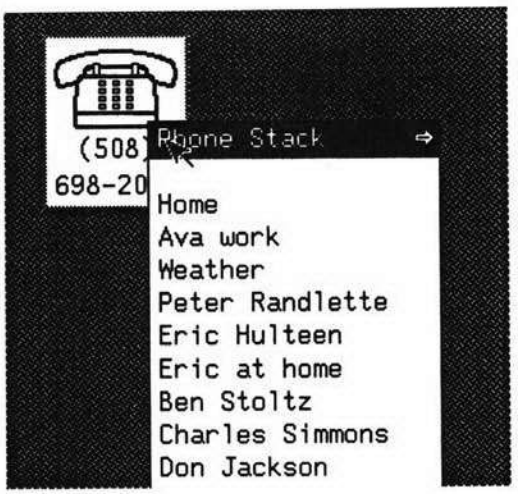

Figure 11.18. The Xphone speed-dial pop up menu. Selecting an entry places a call to the associated number.

An associated name, address, and phone number database tool, Xrolo, provides several other means of dialing. As with Xphone, Xrolo occupies limited screen space until activated, which is done either by typing a name into the small window or by holding down a mouse button to pop up a menu of names (see Figure 11.20). Once chosen, a card appears with relevant information in various fields, each of which can be edited (see Figure 11.21). The search may be based on personal name, computer login, or company. Related applications can generate facsimile cover sheets, long and short address lists, and mailing labels. There is also a text-based version of the tool for use over dialup lines.

Although the lines of text on an exposed card are independent objects and may each be edited individually, the application understands that there is a relationship among them. For example, an address usually consists of a name, a company

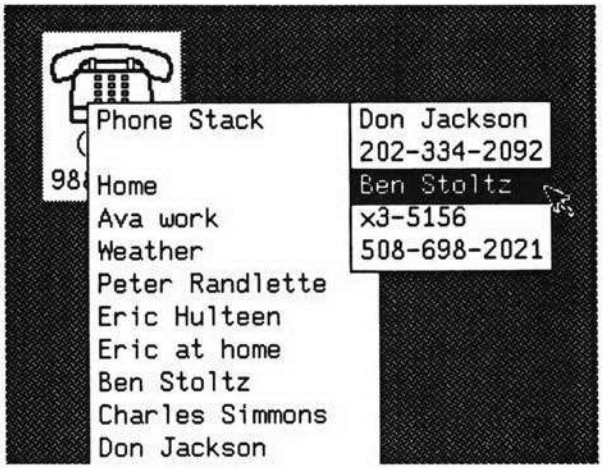

Figure 11.19. The Xphone call-log pop up menu allows recently called numbers to be redialed.

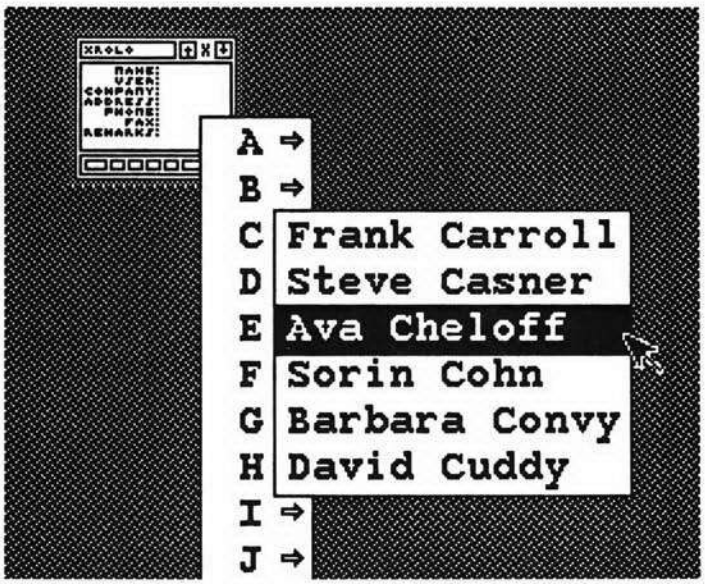

Figure 11.20. Pull down menu of names in Xrolo.

name (if it is a work address), street location, city, state, and postal code. A mouse click on the *work address* button selects all the relevant lines and highlights them appropriately for feedback to the user (see Figure 11.22). This selection may then be deposited by mouse click into another application.

Xphone and Xrolo interact with each other in several ways; the simplest and most common interaction is generating a phone call from the number on a per-

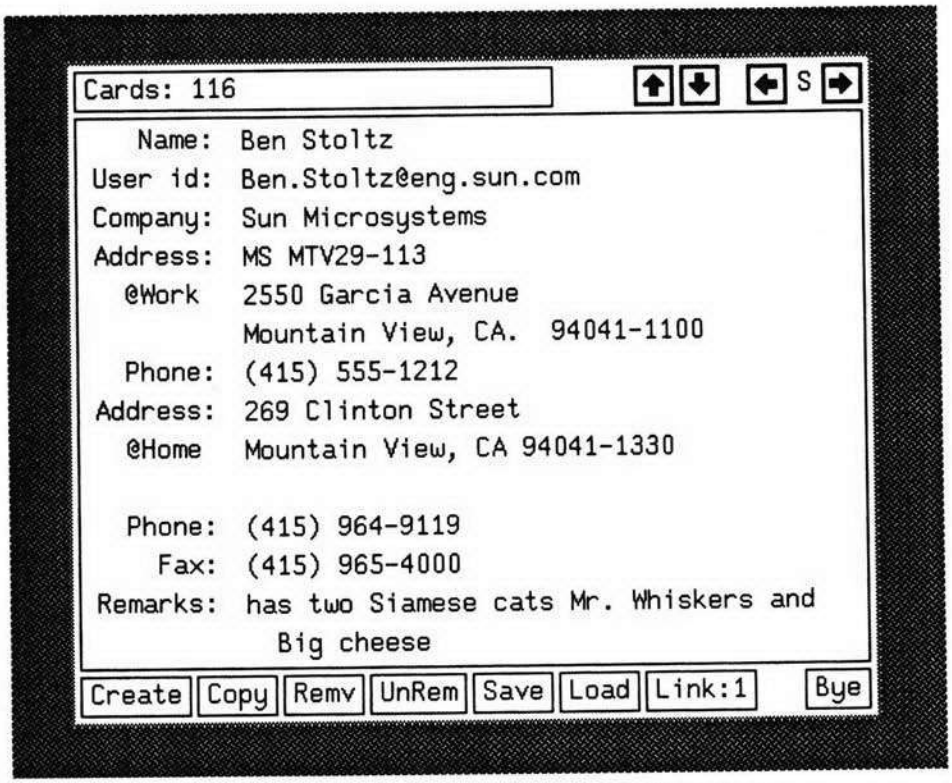

Figure 11.21. A name and address "card" from Xrolo.

son's Xrolo card.[8] Clicking the middle mouse button on a phone number sends a message to Xphone and dials the requested number. A number dialed in this manner would appear as a name in Xphone's phone log menu. A person's name or email address can be stuffed into Xphone just as easily as a phone number, which presents a more involved method of dialing by name. When Xphone parses the string containing a name, it fails to find a valid number and consequently sends a request to Xrolo to search for an entry that matches the string. If a match is found, Xphone obtains the associated telephone number and dials it and the name of the called party is noted in the call log.

Flexible Call Routing

This case study describes a series of interrelated, small-scale Media Lab projects that have a common theme of flexible management of incoming calls. At

[8]Xphone and Xrolo communicate via the X selection mechanism, which provides a rendezvous point and data exchange through the X server.

Name:	**Ben Stoltz**
User id:	Ben.Stoltz@eng.sun.com
Company:	**Sun Microsystems**
Address:	**MS MTV29-113**
@Work	**2550 Garcia Avenue**
	Mountain View, CA. 94041-1100
Phone:	(415) 555-1212
Address:	269 Clinton Street
@Home	Mountain View, CA 94041-1330
Phone:	(415) 964-9119
Fax:	(415) 965-4000
Remarks:	has two Siamese cats Mr. Whiskers and Big cheese

Figure 11.22. Highlighted text of the fields associated with *work address.*

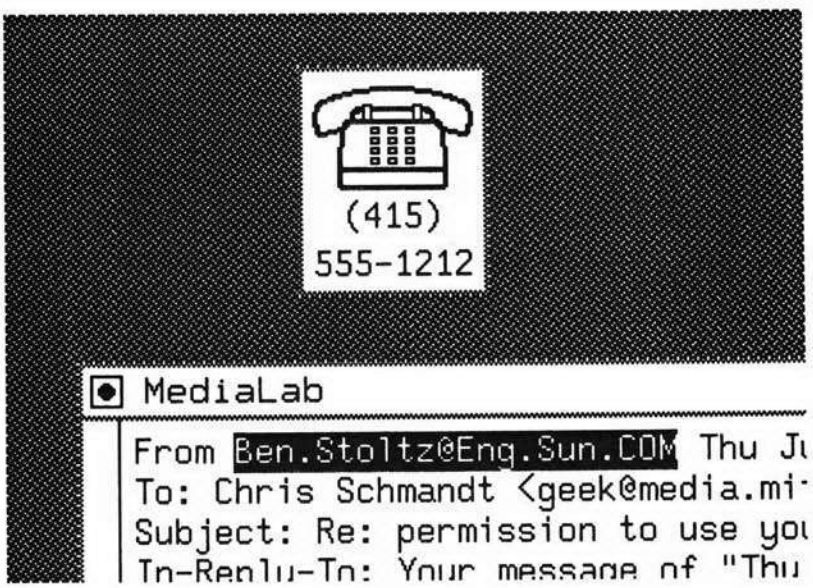

Figure 11.23. Placing a phone call by name.

issue is the potential for computers through their participation in call setup to more effectively connect parties who need to speak to each other. At the same time it is essential to minimize the degree to which users are interrupted by unwanted calls if subscribers are to fully embrace the emerging mobile telephone technologies.

There is little doubt that although nearly all of us view the telephone as an essential part of our lives, we all also experience its annoying aspects as well. We resent unsolicited telemarketing calls at home during dinner or misdialed wrong-number calls in the middle of the night. We are annoyed when a face-to-face conversation in an office is interrupted for a call and are frustrated to wait in line in a hotel lobby while the reception clerk at the front desk takes a message for a guest. When we work alone in our offices, we may resent the interruption of the phone "ringing off the hook" all day; at home we often use answering machines to screen calls (a feature which disappears with centralized voice mail). Yet we rarely disconnect our phones, zealously play phone tag with those we wish to contact, and are disappointed with our few friends who still do not have answering machines.

The popularity of pocket cellular telephones[9] lends credence to the telephone operating companies' dreams of personal communication networks (PCNs) in which we each carry a phone and are always accessible via a universal personal number. Although some intense telephone users are ready for such services today, most of us shy away from the concept of always being reachable. If computers can become involved in call setup, how can they personalize and filter calls to make such services more attractive? This case study discusses some techniques that try to answer these questions based on alternate means of announcing incoming calls, user interfaces allowing manual intervention by either the caller or called party to route calls more effectively, and finally, automatic call filtering and routing by means of a personalized user agent.

The ring of the telephone is pervasive and penetrating, yet conveys only the minimal information that someone dialed a particular telephone number. With the gradual acceptance of calling party identification, consumer devices have become available that display the number of each caller, and many business PBX telephones similarly indicate this information on a small display. Unfortunately, the display built into the telephone is usually difficult to read, and it requires us to remember the phone numbers of all the people we wish to speak with. By the time our gaze is diverted to the telephone set, we have interrupted whatever task we were performing when the call came in. At MIT, telephones may ring differently for on-campus and off-campus calls, but this too is only marginally useful (contrast this with Etherphone's caller-specific ring tunes, for example.) Computer-based visual or audible alerting may minimize the interruption.

[9]In early 1993 the number of cellular telephone subscribers in the United States passed 10 million. There have been suggestions that poorly wired regions of Eastern Europe should bypass the local loop and go directly to universal cellular service.

Alternate alerting at the Media Lab is performed by a variety of workstation programs that act as clients of a centralized telephony server. This server uses a single ISDN interface to send notifications to clients (e.g., incoming call attempts) and can route calls at their request by performing call transfer operations. On-campus calls include calling number information, which the server translates to a name from a database.

One client performs visual alerting, and several others announce calls using synthesized speech. Visual alerting pops up a window on the user's display; as shown in Figure 11.24, this window indicates the caller's name (or "unknown") and disappears when the call is answered or the caller hangs up. Audible alerting in the office consists of a speech synthesizer announcing the caller's name. Additional audible alerting is provided in a large common lab work space where the synthesizer announces both the called and calling parties over a public address system; a similar service was provided under BerBell. Someone with an office near the lab space who is paged may be able to reach the telephone in time to take the call. But a workstation user can also take advantage of a variant on the visual call alert service by sending the call to the nearest telephone. This form of alert window shown in Figure 11.25 includes buttons that will route the call if the user selects one with the mouse.

Finally, some students and staff wear "Active Badges" that include several buttons. The Active Badges from Olivetti Research Laboratory use infrared signals to periodically transmit their identity to a network of sensors in strategic positions around the building [Want and Hopper 1992]. By pressing one of the buttons, a person is also able to route the call to the nearest telephone; this service has the advantage of being accessible even if the called party is not logged on to a workstation. Badges can also receive messages from the sensors to sound a tone or illuminate an LED (this provides yet another means of alerting a mobile

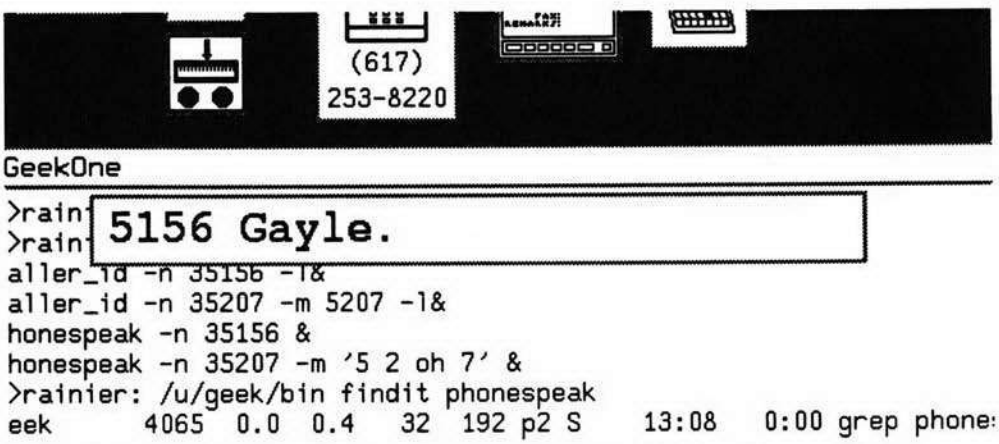

Figure 11.24. Window-based visual call alerting displays the name of the calling party.

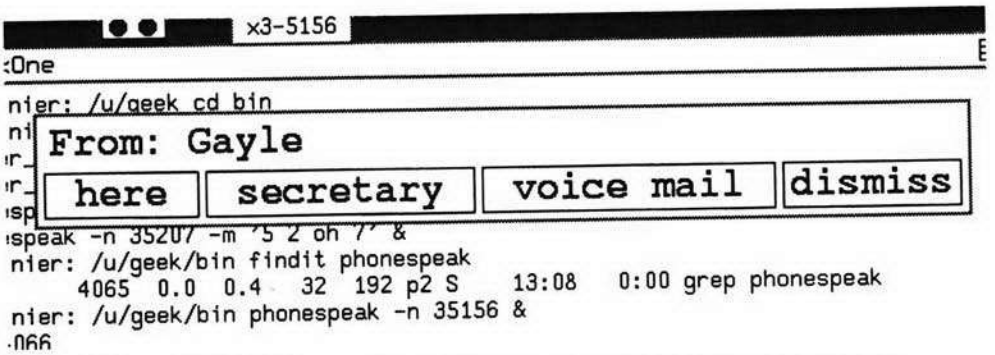

Figure 11.25. When routing buttons are added to the visual alert window, the called party can dynamically forward calls when working in another office.

user to an incoming call). At the Media Lab, an Activity Server combines badge-sighting reports with information about which users are logged in to which workstations on the local network (the "finger" command) to track users' activities and locations even when they are not wearing badges. The Activity Server also receives events from the telephone server and is notified when any lab telephone is in use.

Knowing colleagues' locations and activities helps local lab callers judge whether to interrupt someone with a phone call and know that person's location if they do proceed to call. A graphical user interface, Watcher (shown in Figure 11.26) displays users' physical locations and states, e.g., "alone in own office," "on the phone," or "in a meeting" (with other badge wearers) along with the number of the nearest phone. Clicking on the *phone* button sends a message to Xphone causing it to call the displayed number.

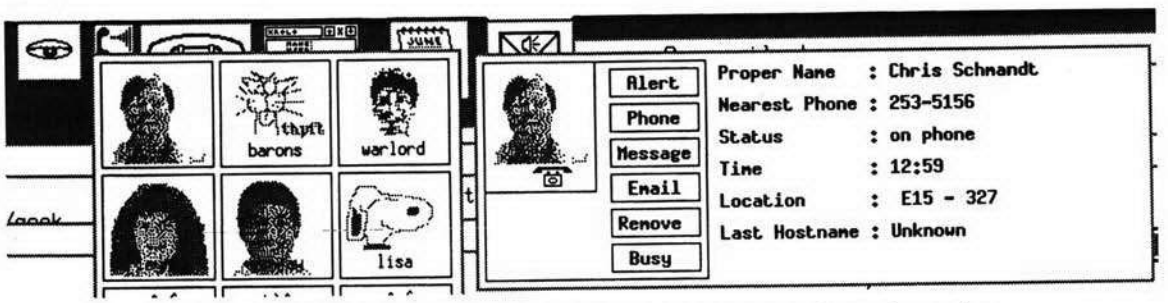

Figure 11.26. *Watcher* shows the location and activity of members of a workgroup. Buttons in the detail windows choose different means of sending messages to the selected person.

An interface such as Watcher helps the caller decide whether and where to phone, but it is available only to someone who is calling from a workstation in the lab.[10] The flexible alerting and call forwarding mechanisms just discussed enable the called party to manually intervene in call routing in response to the stimulus of an incoming call but depend on the availability of additional hardware such as a nearby workstation screen or a badge. A similar style of hardware-based remote intervention using pager technology is described in [Kramer *et al.* 1993]. Ideally one's personal workstation would make routing decisions automatically based on knowledge of the identity of the caller, the time of day, the location and activity of the called party, and whether a secretary or receptionist was available to take the call if it otherwise went unanswered. A person's location and activity may be derived from the dynamic information gathered by an entity such as the Activity Server, but personal calendars also give clues as to travel plans and meetings scheduled in advance.

A Media Lab thesis project [Wong 1991] explored the feasibility of call forwarding based on user-authored rules conditioned on the routing factors just mentioned. The graphical user interface shown in Figure 11.27 was used to create rules. A rule was defined by clicking with the mouse on icons representing four classes of conditions shown on the left of the figure; at this point, text-based dialog boxes appeared that allowed the user to complete the condition, e.g., specifying the time of day or name of the calling party. Each rule routed calls to destinations depicted on the right; these could be either a predefined phone number (e.g., a secretary's desk), voice mail, or a more dynamic location (e.g., the nearest available telephone). A knowledge-based "electronic receptionist" (from Bellcore) without a graphical interface is described in [Gifford and Turock 1992], which presents usage data supporting the position that automatic routing of some telephone calls may provide a valuable service.

Although initially promising, this method of defining call routing actions was problematic for several reasons. The graphical user interface was designed to shield users from the syntax and other details of the text-based rule language used to specify call routing. Although it was fairly effective for creating rules, the interface turned out to be unwieldy for modifying or deleting rules; it was difficult to graphically display a rule, especially one with multiple conditions for activation. The graphical interface did not provide capability for testing and debugging rule sets.

It may also be difficult for users to match their mental models of how the phone should behave with the operation of such an explicitly rule-based approach. Rules may conflict. For example, if I have one rule that says that after 5 P.M. calls forward to voice mail and another which says that calls from my daughter always forward to the nearest phone, then what should happen when my daughter calls

[10]Actually through the Phoneshell service described in Chapter 12, a knowledgeable subscriber can phone in, discover where other users are located, and then transfer the call to that location.

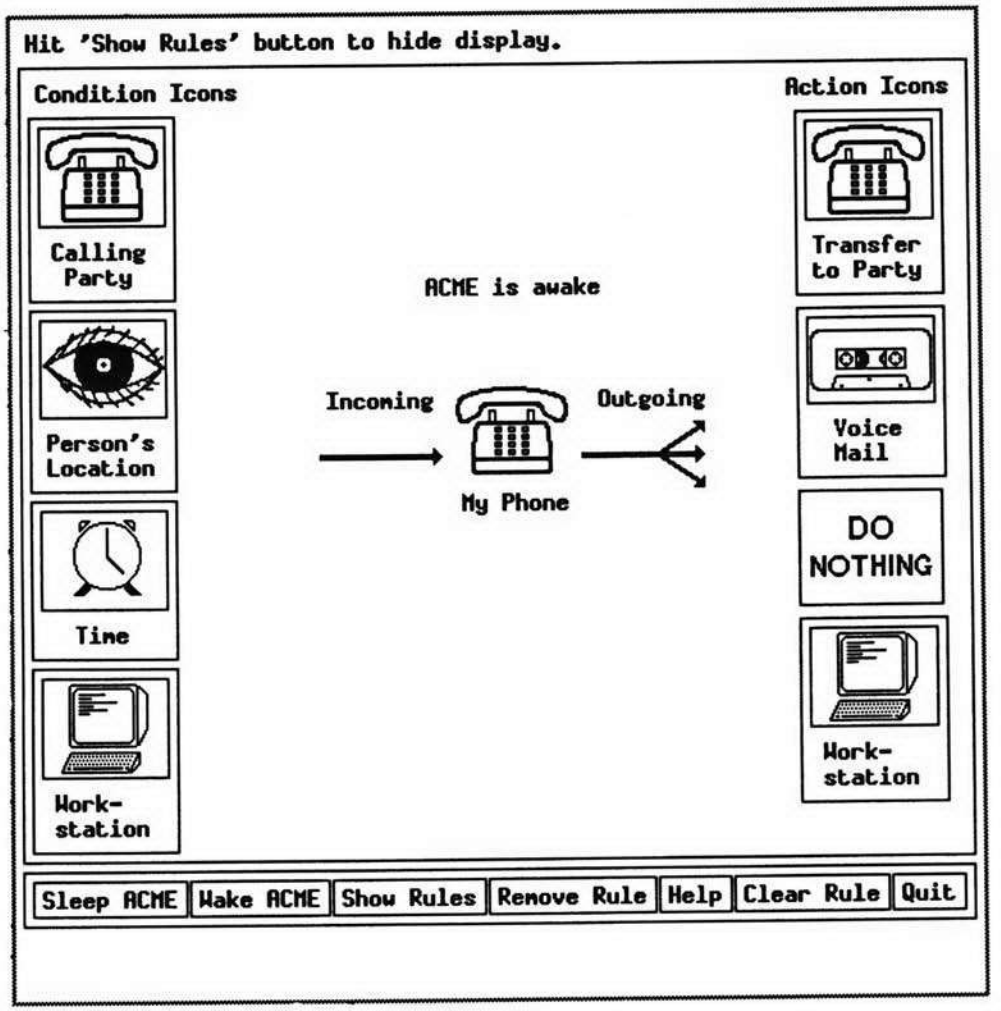

Figure 11.27. The graphical user interface to a rule-based call router. The user selected conditions on the left and associated them with routing destinations on the right.

at 5:30? This might be resolved by saying that rules involving the identity of the calling party take precedence over rules based on other conditions as telephone calls are more highly associated with the person calling than the time of day, for example. But what should happen if I also have another rule that says that I am never to be disturbed when in the conference room and when my daughter calls at 5:30 that is my location? Flexible call routing may be better performed by software that learns my preferences by observing my behavior over time, and this requires a feedback mechanism to retrain such software whenever it makes an incorrect decision.

SUMMARY

This chapter was about computers performing telephone management tasks. Computers can place calls on behalf of a user, answer the telephone, or decide how to route an incoming call. This chapter began by making the case for computer-mediated telephony on the basis of its providing superior user interfaces than telephone sets, facilitating a more integrated overall electronic communications perspective, and extending the variety of call management services to which users may subscribe.

To place, receive, and route telephone calls, computers require hardware interfaces capable of communicating with the telephone switch. In some circumstances it may be appropriate to place such hardware on each computer in every office; this is a fully-distributed architecture. If telephone interface hardware is excessively costly or not compatible with the workstation in each office, a centralized approach may be preferable. With a centralized architecture, a single server computer with appropriate hardware manages telephones for many client computers. In either hardware scheme, centralized or distributed, a software-based server approach allows multiple client applications to participate in telephone management and effectively insulates application software from the details of the physical arrangement of hardware.

The concept of computer management of telephone calls was illustrated by a number of example projects discussed in different levels of detail. Etherphone and PX utilized distributed architectures to control flexible call switching entities, while MICE and BerBell explored the potential for client-based call control using centralized hardware similar to a telephone company's central office switch. Phone Slave presented a unified call management user interface including sophisticated answering machine capabilities. Xphone and Xrolo are examples of more modern software architectures and illustrate how multiple applications may coordinate with each other to provide enhanced access to multiple communication functions. Finally, a variety of small applications were discussed together under the theme of computer routing of incoming calls, either manually or automatically, and with or without the cooperation of the calling party.

The variety of functionality computers may bring to telephone management motivates and underscores the research projects discussed in this chapter. Some examples included enhanced message-taking features as well; once messages have been recorded, computer users can benefit from improved user interfaces to voice mail and may use these messages in other applications as well. This theme is continued in the next chapter, which discusses uses and interfaces for managing stored voice on the desktop.