# PHONETOOL: INTEGRATING TELEPHONES AND WORKSTATIONS

Chris Schmandt
Media Laboratory — Massachusetts Institute of Technology

Stephen Casner
Information Sciences Institute — University of Southern California

## Abstract

In this paper we argue for computer workstation access to voice telephony functions. We describe several computer based communication tools developed in our laboratories, and explore one, *Phonetool*, as a detailed example. We discuss user interface issues for collections of such utilities. Finally, we discuss several architectures for interaction with the PBX and the communication requirements between workstation and switch.

This paper espouses the need for computer workstations to control conventional telephony functions, concentrating on a graphical dialing tool as an example. By way of this example, we will explore several architectures for software and hardware access to PBX equipment. An underlying motivation for this work is a vision of the workstation as an integrated and personalized communications node, capable of interacting with a variety of media and transport mechanisms.

Many portions of this vision are already in place, but not integrated. Users store phone numbers in on-line databases. Computers are being used to send and receive facsimile, including fax/e-mail gateway services. Voice mail systems are computers that intercept calls and digitize speech, though they don't yet provide a means to access the stored voice from the rest of the office computer environment. A greater variety of services (conference, forward, transfer, pickup, etc.) are becoming more widely available on digital PBXs; the addition of control interfaces on PBXs opens the possibility for workstations to perform intelligent call setup and routing using these services.

Our position argues that as office computer usage increases, greater functionality is gained from allowing computers access to telephone switches than by building more complex telephone sets. This view is not unique to the authors. The MICE project at Bellcore [2] emphasized user interface and personalization of services in a network testbed environment. BerBell, also at Bellcore [3] built a programmatic interface to several switches. Etherphone [8, 9] at Xerox PARC explored alternate transport mechanisms and expanded use of voice in office workstations. Though less ambitious (it is a product, after all) Northern Telecom's Meridian Mail gives networked personal computer users integrated graphical access to voice and text messages. DEC's *Computer Integrated Telephony, CIT* [7], defines a model for linking telecommunications and computing systems for functional integration. Current work at Olivetti Research Center [1] focuses on developing software architectures for sharing voice and telephone functions between multiple processes and applications.

## 1 Computers or Telephones?

Why do we want to interface telephony functions to computer workstations? The reasons fall into two broad categories: improved user interfaces and enhanced functionality.

Workstations are already much better suited than telephones for improved user interfaces. They have far superior displays, typically CRT screens; in part because of power requirements and cost, telephone displays are small and difficult to read. Workstations have better input devices, with full keyboards and the ubiquitous mouse. Most important, a variety of powerful window systems provide rich and flexible graphical interfaces (see figure 1).

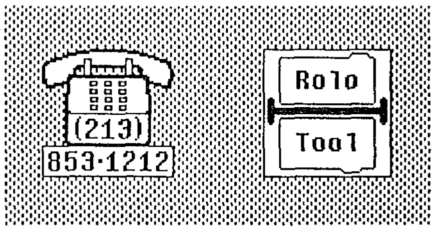


Figure 1: "Iconic" form of our tools

These user interfaces may be customized; recent window systems provide ample opportunity for users to configure the mapping from input to function, as well as appearance (color, font, etc.). There is every reason to extend this personalization to telephony functions, such as:

- how calls get routed as a function of time of day, calling party ID, personal schedule, etc.
- the names and number of entries in a speed dialing menu
- voice mail prompts and interaction with frequent versus casual callers

Such personalization is difficult to control with the traditional 12 button telephone interface. We are already at the point where new telephone functionality is infrequently used because it is too difficult to access.

Personalization may be difficult to implement in switch software, with its emphasis on state tables and real-time requirements. At this stage in the design of user interfaces, we require flexible operating systems and good software development environments. Design will be iterative and interfaces experimental for some time. Since current workstations already include window systems, there is no need to duplicate that resource elsewhere. Even a simple resource such as memory is more effectively used when it can be shared with a number of applications, and allocated according to moment-to-moment needs.

Enhanced functionality is the result of new services and, most important, integration at various levels. Integration comes in many forms, such as:

- linking databases to functions, e.g., dialing from an on-line directory
- a communications orientation of many computer-accessible media (electronic mail, facsimile, modems)
- increasing potential for use of new media, such as voice in computers and video in telecommunications

At least three elements are needed for these new media to be useful: transport, storage, and user interface. By combining telephony with the workstation to achieve all three, we envision a scenario such as the following:

> While out of my office I receive a call from my co-author; a voice message is taken. On my return, I use the mouse to play (and repeat) portions of the message, then I call him back via my speed dialing tool, using calling party ID if available or his phone number from an on-line rolodex. While on the phone, we both listen to pieces of his previously recorded message, under my control, again with my mouse.

## 2 Applications

In this section we will describe samples of some of the pieces required to make the above scenario a reality. The section following will expand on computer to PBX communication requirements to support such systems.

### 2.1 Phone Slave

Phone Slave was a combination of answering machine and voice mail system providing a graphical user interface to messages. It has been described in detail elsewhere [4, 5]; we mention it here very briefly as an example.

Phone Slave greeted the caller with digitized voice, playing a series of prompts and recording answers. Since calling party ID was not available, it attempted to use speech recognition to identify known callers, who would then receive a personal message. It included speed dialing from a database. It allowed local and remote access to both voice and text messages (using text-to-speech synthesis).

Although the Phone Slave process was usually idle waiting for ring, responsiveness was critical once a call was answered, especially in transitions between play and record operations. Playback of voice messages was also tightly coupled with the workstation display and user interface. Synchronization between audio data and user input events allowed interruptibility as well as predictable repetition of a particular sound segment.

### 2.2 Phonetool

Phonetool provides a variety of methods to automate telephone dialing from a workstation. Unlike most window-based programs, Phonetool remains active when its window is closed as the "icon" pictured on the left in figure 1. The small size of the icon keeps it out of the way but handy to place a phone call while other work is being done in other windows. Phonetool functions are invoked with the buttons on the mouse while the cursor is on the icon:

- A phone number appearing in any window on the screen can be dialed with just a few clicks of the mouse buttons. After selecting the characters of the phone number, a click of the left mouse button on the Phonetool icon will dial the number. This mechanism is particularly convenient for returning calls when phone messages are sent via e-mail.
- The middle mouse button redials the previous phone number (the one displayed in the icon).
- The right mouse button displays the the "Call" submenu, a personal list of people or places to call. Choosing one of these names will cause the corresponding number to be dialed (figure 2).

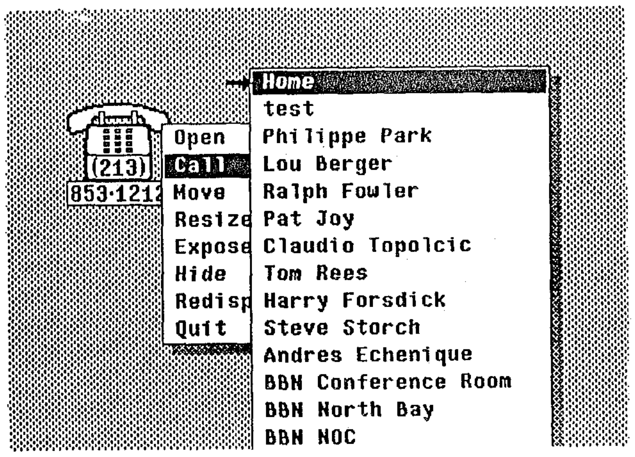


Figure 2: *Phonetool*, icon and menus

Phone numbers can be dialed directly into the Phonetool icon using the digit keys or the numeric keypad. This method offers some advantages over dialing on the phone directly:

- The number being dialed is displayed in the icon and saved for redialing.
- Mistakes can be corrected with the DELETE key.
- There is no time limit between digits.

When open, the Phonetool displays in one subwindow a log of calls placed. A second subwindow may be used as a scrollable phone number directory when there are too many names for practical inclusion in the "Call" menu. The directory may be scanned visually or by automatic search for a name typed in. Alternatively, structured storage of a complete address and phone number directory is provided by Rolotool, described next.

## 2.3 Rolotool

Rolotool is an example of the inescapable database behind all our communications. Inspired in part by Phonetool's conservative use of screen space, Rolotool usually appears as a small icon. Using the mouse, one can expose pop-up menus of the alphabet with pull-right menus of names (see Figure 3). When a menu item is chosen, a "card" of information appears for reading or editing (see Figure 4). The tool is also sensitive to the keyboard; when the mouse cursor is inside the icon, simply typing part of the name, user ID, or company name of the desired party will cause the corresponding card to be displayed.

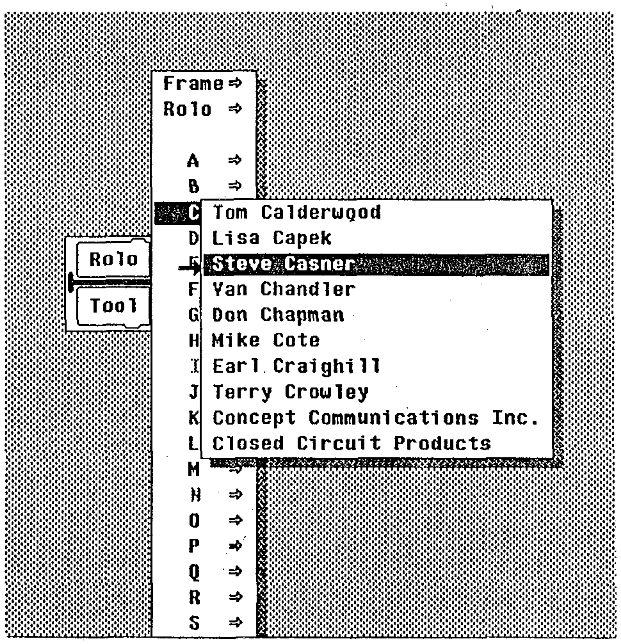
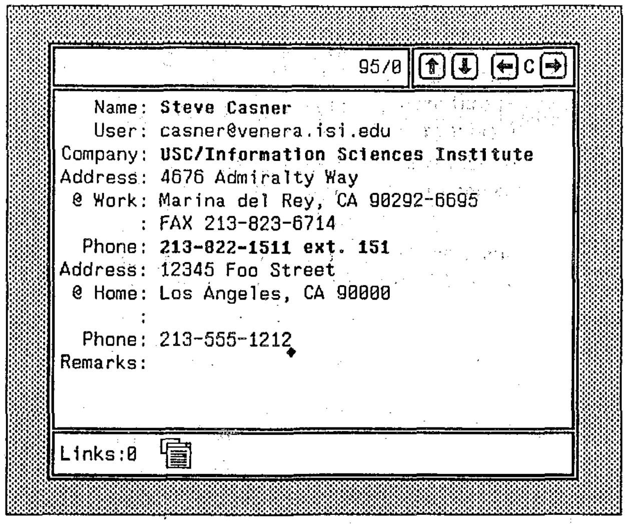


Figure 3: *Rolotool's* cascading menus

Rolotool is meant to work in concert with other applications, via the cut/paste ("selection") mechanism found in all window systems. For example, the user may click on the field "address at work"; this will make the window system selection hold the correct information, i.e., name, company, and street address and the card gives feedback by bold-facing those fields. This selection may be sent to a file for printing, i.e., a mailing label, or perhaps stuffed into an e-mail message.

An implicit selection mechanism links Rolotool and Phonetool so that phone numbers selected on the Rolotool card are automatically dialed through the Phonetool.



```
                                   95/0
    Name: Steve Casner
    User: casner@venera.isi.edu
 Company: USC/Information Sciences Institute
 Address: 4676 Admiralty Way
  @ Work: Marina del Rey, CA 90292-6695
        : FAX 213-823-6714
   Phone: 213-822-1511 ext. 151
 Address: 12345 Foo Street
  @ Home: Los Angeles, CA 90000
        :
   Phone: 213-555-1212
 Remarks:

Links:0
```

Figure 4: A Rolotool directory "card"

## 3 Computer - Switch Communication

To implement the functions of these tools and to enhance them with more sophisticated telephony tasks, various interactions between the workstation and telephone switch are needed. The more detailed such communication is, the more sophisticated interface we can build on the workstation, and the more advanced features we can program.

Calling party ID is perhaps the most useful piece of information to communicate about an incoming call. It can enable intelligent voice messaging, including delivery of personal messages to known callers. It is useful for "callback" features, in which the calling number can be left with a busy destination to allow the busy party to return the call on completion. One of the most powerful uses of calling party ID is for personalized call routing; some known callers may be routed through to interrupt a meeting, for example, while a message will be taken for others.

Call progress information is also quite useful. The Phonetool can be more helpful at deciding how to handle an incomplete call if it can differentiate between busy and ringback-no-answer. Call completion is useful for a message system trying to deliver stored voice messages.

For personalized call handling we need a set of routing primitives available via the PBX. As an example of the desired behavior, consider time-dependent call routing. After hours, an incoming call from home should perhaps ring in two locations, office and lab, where the lab line is not a simple extension of the office line during normal hours. If neither line answers, a voice message should be taken, perhaps illuminating both message waiting lights, but requiring a password to hear the message from the more public phone.

To implement the above scenario, we would like to "claim" the connection, cause both numbers to ring, and decide whether to record a message depending on whether any number is picked up. More likely, we will have to answer the incoming call, ring the second number, and transfer on answer. Perhaps the calling party could be conferenced in to give audible ringback.

Current switching functions are derived from historical developments rather than current needs or the desire to get the routing abstractions "right". Perhaps the switching functions can be rebuilt from more basic primitives that may then be combined into new, more powerful functions.

Many of these features are more likely to be available on calls internal to a PBX than from the outside, but we would still like to take advantage of them. Calling party ID is a particularly controversial subject; even when the technology is available, popular opinion may mandate against it. However, with the ISDN switch recently installed on the MIT campus, many people have found the calling party ID on internal calls to be useful in deciding whether to answer a call and even how a caller is greeted.

We desire the communication function between workstation and PBX to be available to all workstations to allow fully distributed user interfaces. However, it may be convenient to use a hardware architecture in which the workstations communicate over a (computer) network to a single device linked directly to the switch. We discuss these alternatives in the following section.

## 4 Architectures

Perhaps the most obvious method for controlling a telephone from a workstation is to wire the two directly together with a suitable interface. However, unless the telephone is designed with an interface for remote control that is compatible with an existing interface on the workstation, some modifications to the telephone and/or additional hardware will be required for each installation.[1]

The connection can be as simple as hooking an autodial modem into the phone line or as complex as adding a digital signal processor board with a telephone line interface for complete control of signalling and telephone (even speakerphone) functions. This method has the advantage that it will work with a standard analog phone line and places no special requirements on the switch, but little communication is received back from the switch. Since many PBXs now use (proprietary) digital telephone sets, this method is not as universal as it once was, but emerging ISDN interfaces and products will again provide standardization plus enhanced functionality.

In a setting where there are many workstations connected by some form of network and the telephones are switched by a PBX, a centralized architecture can be used to avoid the cost of replicating hardware at each installation. The workstations communicate over the network to a central server which in turn communicates with the PBX through a single interface. It is possible to implement such a server with nothing more than standard analog connections to the PBX and using features that are common in modern PBXs, but functionality will be limited. With digital telephones and a sophisticated control connection from the server to the PBX, this centralized architecture could provide all the functionality of per-workstation interfaces and probably more. This is the model for CIT [7].

Example implementations for each of these architectures are described in the following sections.

[1]An installation of this type was made by Danny Cohen at ISI in 1980. A computer terminal was interfaced into a modified speakerphone through a microprocessor with relay outputs. This was the inspiration for development of the Phoneserver (see section 4.1).

### 4.1 Centralized Architecture

A centralized server called "Phoneserver" has been in operation at ISI since 1987 to provide automated call placement functions. The underlying hardware is a Rolm CBX telephone switch with a digital ROLMphone and a workstation in each office. The workstations are all connected by a 10 Mb/s Ethernet LAN. The Phoneserver program runs on one of the workstations and is connected through a digital interface to a special ROLMphone called a Juniper. A diagram of the configuration is shown in figure 5.

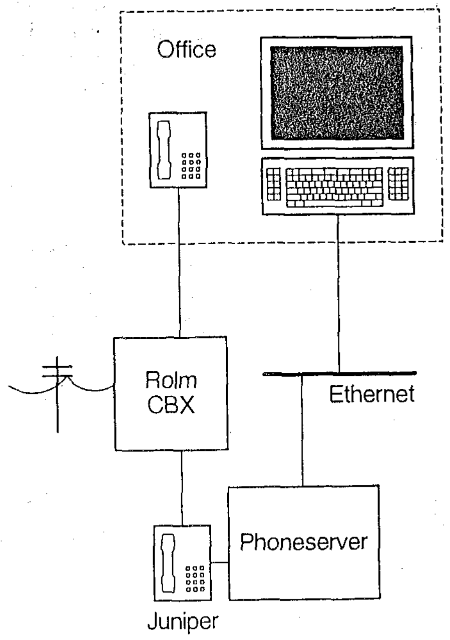


Figure 5: Centralized *Phoneserver* architecture

Call placement requests are generated through a variety of user interface programs, such as Phonetool, running on the users' workstations. Each request is sent as a single packet over the LAN to the Phoneserver. The request includes the extension number of the telephone next to the workstation plus the number to be dialed. The Phoneserver sends an acknowledgment packet back to the workstation and then begins dialing though the Juniper telephone. Request processing takes four steps:

1. An "intercom call" is placed from the Juniper to the requester's telephone. This type of call does not ring on the called phone; it just gives a two-tone sound and completes the connection without the called party (the requester) having to touch the phone.
2. The intercom call is put on "soft hold" while a call is placed from the Juniper phone to the requested phone number.
3. The requester's telephone is connected up to the second call by "conferencing" the two calls together.
4. The Juniper phone drops out of the call to be ready to service the next request.

The Phoneserver uses only standard features of the Rolm CBX and the digital ROLMphones. Each person can use this service without any additional hardware; only software changes are required to add a new user. The new user's telephone extension must be configured into the Juniper phone's "COM group" to allow the intercom call to be placed. In addition the user must specify in a configuration file on the workstation the extension number of the associated telephone, to be used in dialing requests to be sent to the Phoneserver.

The Phoneserver could be implemented with an analog interface and tone signalling instead of the Juniper phone. Dialing through the Juniper is faster and more robust because the commands and status responses are explicit character codes on a digital interface. The Rolm CBX provides more status information on the Juniper interface than call progress tones would give.

It would also be possible to use analog telephones at the workstations, but then the phone would ring and have to be answered before the server would dial the requested number. This was a drawback to MICE, which used analog phones with a centralized server (one computer on the network communicated directly to the switch, a Redcomm MSP, over a serial data line).

### 4.2 Distributed architectures

When Phonetool was imported to MIT, which at the time had analog Centrex, the Phoneserver was replaced with a simple distributed architecture. A modem in parallel with the telephone placed the call. The modem speaker was turned on to allow call progress monitoring, and the user took the phone offhook to talk (the modem would time out and hang up shortly thereafter).

Later, a very similar interface was built to a voice server [6]. A PC with one of several voice/telephone boards and a single line was dedicated to each workstation. The real power of the voice server is its ability to answer calls, digitize messages, and monitor call progress better. By connecting a speakerphone, always off-hook, and using the board to control hook state, hands free calling was again possible.

Etherphone is an example of a hybrid architecture. Special equipment to interface to the telephone line and Ethernet is installed in each office, although some services, such as voice storage and call routing, are centralized. Both voice and control signals use the same Ethernet cable for transmission. The workstation user interface communicates with the centralized server, which talks to the distributed hardware (often next to the original workstation).

## 5 Conclusions and ongoing work

We have argued for increased use of computer workstations for voice telephony, as opposed to more sophisticated telephone terminals. Telecom equipment manufacturers tend to react to modern needs for improved telephony by building larger and more sophisticated telephones, and adding hosts of features to their switches. Although some of these telephones and services are quite impressive, they will not replace computer workstations in the office.

Furthermore, we claim that the workstation is a much more effective place for the telephony user interface and personalized functionality. We have offered several examples of workstation applications supporting various aspects of telephony. The degree to which the user interface can be implemented on a workstation depends entirely on the functionality provided by switch vendors to allow communication with their equipment. We detailed the implementation of Phonetool to offer examples of communications needs and possible hardware architectures to implement such interaction.

At MIT, the three tools described in this paper are being ported to the X Window System, and more will be added. We are exploring the use of voice in multi-media inter-client communications under this window system. We will shortly be using the Olivetti VOX audio server [1], which provides network access allowing for distributed interfaces but centralization of some resources as appropriate. Finally, ISDN interface hardware will be incorporated to communicate with MIT's 5ESS switch.

## References

[1] B. Arons, C. Binding, C. Schmandt, and K. Lantz. The VOX audio server. In *Proceedings of the 2nd IEEE Comsoc International Multimedia Communications Workshop*, Ottawa, Ontario, April 1989.

[2] G. Herman, M. Ordun, C. Riley, and L. Woodbury. The Modular Integrated Communications Environment (MICE): A system for prototyping and evaluating communications services. In *Proceedings of the 1987 International Switching Symposium*, pages 442–447, 1987.

[3] B. E. Redman. A user programmable telephone switch. Technical Report TM-ARH-009118, Bellcore, 1987.

[4] C. Schmandt and B. Arons. A conversational telephone messaging system. *IEEE Trans. on Consumer Electr.*, CE-30(3):xxi–xxiv, 1984.

[5] C. Schmandt, B. Arons, and C. Simmons. Voice interaction in an integrated office and telecommunications environment. In *Proceedings of the AVIOS '85 Voice Input/Output Systems Applications. Conf.*, 1985.

[6] C. Schmandt and M. McKenna. An audio and telephone server for multi-media workstations. In *Proceedings of the 2nd IEEE Conference on Computer Workstations*, pages 150–159, Santa Clara, CA, March 1988.

[7] C. Strathmeyer. Voice/data integration: An applications perspective. *IEEE Communications Magazine*, 25(12):30–35, 1987.

[8] D. Swinehart, L. Stewart, and S. Ornstein. Adding voice to an office computer network. In *Proceedings of GlobeCom '83, IEEE Communications Society Conference*, pages 392–402, November 1983.

[9] P. Zellweger, D. Terry, and D. Swinehart. An overview of the Etherphone system and its applications. In *Proceedings of the 2nd IEEE Conference on Computer Workstations*, pages 160–168, Santa Clara, CA, March 1988.