A low-power, lightweight unit to provide ubiquitous information access

Application and Network Support for InfoPad

Shankar Narayanaswamy, Srinivasan Seshan, Elan Amir, Eric Brewer, Robert W. Brodersen, Frederick Burghardt, Andrew Burstein, Yuan-Chi Chang, Armando Fox, Jeffrey M. Gilbert, Richard Han, Randy H. Katz, Allan C. Long, David G. Messerschmitt, and Jan M. Rabaey

Some of the most important recent trends in computer systems are the emerging use of multimedia Internet services, the popularity of portable computing, and the development of wireless data communications. The primary goal of the InfoPad project is to combine these trends to create a system that provides ubiquitous information access. The system is built around a low-power, lightweight wireless multimedia terminal [1] that operates in indoor environments and supports a high density of users. The InfoPad system uses a number of innovative techniques to provide the high-bandwidth connectivity, portability, and user interface needed for this environment. This article describes the design, implementation, and evaluation of the software network and application services that support the InfoPad terminal.

We believe that future users will demand network connectivity in their mobile computer systems. They will want access to documents, weather reports, traffic information, stock prices, bank accounts, maps, and other documents that reside on the Internet or on their own local area networks (LANs). There are currently several alternatives that provide this connectivity to the mobile user. A cellular telephone modem could provide mobile data communication, but the low bandwidth and quality of cellular links severely limit the types of multimedia information that can be delivered. For example, National Television System Committee (NTSC) quality video requires approximately 1.5 Mb/s, which is about two orders of magnitude higher than cellular modem data rates. Through the use of a pico-cell networking architecture and wideband spread spectrum radios, the InfoPad system aims to provide up to 2 Mb/s of bandwidth to each of 50 users in an area the size of a typical classroom.

Portable applications often demand hands-free or single-hand operation, which does not fit well with the traditional keyboard and mouse user interface. As a result, many recent portable devices have developed pen-based interfaces (some with handwriting recognition) as a replacement. The primary weakness of these implementations is the low accuracy of the handwriting recognition algorithms that are supported by the portable units. This accuracy can be improved by moving the computation required for recognition into the backbone network, thereby allowing more compute-intensive algorithms to be employed. The InfoPad design also uses speech recognition, video, and audio to enhance usability. Since powerful network-based computation is used, the limitations of portable operation do not force compromises in the sophistication and performance of the user interface.

The demand for portability is clearly demonstrated by the rapid growth in sales of equipment such as portable computers. The main limitation of current portable computers is short battery life. Unfortunately, battery technology is improving at the relatively slow rate of 30 percent every five years [2] with no breakthrough technologies expected. The typical approach to improving battery life is to re-engineer chips and systems for lower-voltage operation. Beyond exploiting this approach in the InfoPad system, a much more dramatic step is being taken by eliminating as much as possible of the electronics and local computation in the portable. This is done by migrating the power-hungry tasks that typically occur in a portable terminal to servers on the backbone network. This approach allows the design of a very inexpensive, lightweight terminal with a long battery life.

Based on these requirements of connectivity, portability, and usability, we designed the InfoPad system around a simple portable multimedia terminal. Although the portable terminal contains no general-purpose computational resources, it does contain a sophisticated low-power chipset that displays text and graphics, plays audio and compressed video, records audio, and captures pen input. The pad also contains a high-speed wireless network interface that connects it to a wired backbone network. All computation needed to make the pad more than a display device occurs on servers that reside on the wired backbone. These servers transmit data such as screen updates, video streams, and audio samples across the wireless link to the pad. The design's main advantages are resistance to errors in wireless transmission, improved portable battery life, and better access to computational power for applications.

The rest of the article is organized as follows. In the following section, we provide an overview of the system architecture. The third section describes the hardware briefly to provide a basis for understanding the software architecture. The software architecture is described in greater detail in the fourth, fifth, and sixth sections. The seventh section presents the performance of the system and some of its components. We discuss directions for future research in the eighth section and conclude with a summary of lessons learned in the final section.

This work was supported by ARPA, IBM, Motorola, HP, and GE/Ericsson.

 1070-9916/96/$05.00 © 1996 IEEE

System Overview

The InfoPad system is composed of the following four layers:

- *Pad*: A low-power, portable multimedia terminal capable of displaying text and graphics, playing audio and compressed video, recording audio, and capturing pen input
- *InfoNet*: A network management and routing system that supports mobility and distributed processing
- *Type servers*: A set of network-transparent servers that shield applications from knowledge of the mobile environment and terminal hardware
- *Applications*: A set of applications optimized for use on the pad

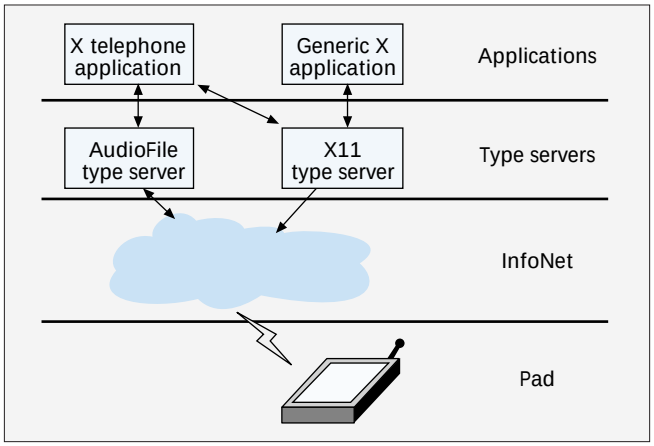
This layering is shown in Fig. 1. The pad, InfoNet, type servers, and applications are described in the third, fourth, fifth, and sixth sections, respectively.

The hardware in the pad supports the operation of the wireless link, and the acquisition and display of multimedia data. The processing in the terminal is primarily performed by a set of application-specific integrated circuits (ASICs) designed specifically for low power dissipation [1]. The embedded software on the pad currently executes on a combination of a low-power processor (ARM 610) and special-purpose programmable logic. This software initializes and controls the ASICs, implements wireless link-level and reliable transport protocols, and monitors the quality of the wireless channel.

The InfoNet layer presents the software layer above it, the type servers, with an abstraction in which each pad appears as a stationary, network-connected multimedia terminal. This layer contains the routing algorithms to make mobility seamless and the routines to manage the wireless network resources (e.g., allocation of frequencies to pads).

The type servers make the pad appear as a typical workstation to applications; this allows compatibility with standard workstation software. A network-transparent server was developed to simulate each of the devices found on a workstation (e.g., display, keyboard, mouse, audio, etc.). For example, an X11 server supports the display of text and graphics on the InfoPad. Standard X clients can connect to this server to display windows without modification. All servers attempt to follow existing standards as much as possible.

Although standard workstation applications can be run on the InfoPad, they do not take advantage of the special characteristics of the pad. These special characteristics include small screen size, lack of a keyboard, and support for handwriting and speech recognition. For this purpose new applications were developed which attempt to address these limitations. These applications make effective and efficient use of recognition and develop new paradigms for using pen and audio.



■ **Figure 1.** *InfoPad system layering.*

The Pad

To perform its function, the pad must sustain sufficient wireless bandwidth for the transport of video, graphics, audio, and pen information while consuming as little power as possible.

The different data types supported by the pad require different amounts of radio link bandwidth. Pen data requires 4 kb/s on the radio uplink (pad to wired backbone), audio data requires 64 kb/s on both the uplink and downlink (wired backbone to pad), video needs 600 kb/s on the downlink, and text and graphics use a variable data rate on the downlink. The radio link is therefore asymmetric, with a much higher bandwidth required on the downlink. The current wireless link implementation uses a frequency-hopped 625 kb/s radio for the downlink and a direct sequence spread-spectrum 242 kb/s radio for the uplink. In the near future the downlink radio will be replaced by a newly designed 2-Mb/s 50-channel custom code division multiple access (CDMA) radio [3] and the uplink by a time division multiple access (TDMA) radio loosely based on the Digital European Cordless Telecommunications (DECT) protocol [4].

We approach the crucial problem of power consumption at the circuit, algorithmic, and architectural levels. The custom ASICs on the pad use a low-power very large scale integration (VLSI) cell library designed to operate at 1.5 V rather than the usual 3.3 or 5 V found in most commercial systems. Algorithms that can be implemented with low power consumption were chosen for the pad. For example, vector quantization (VQ) is used for video compression, since decompression consists of simple table lookups. As mentioned previously, we also significantly reduce power consumption by migrating computationally intensive tasks from the pad to the backbone network. The pad only contains the hardware necessary to control the communication link, display video and graphics, record and play audio, and support pen input on the pad.

The current terminal consumes 175 mW at 5 V for the custom chips, 2.4 W for programmable logic and microprocessor, 2.1 W for the 640 x 480 monochrome LCD display (primarily the backlight), 3 W for the 240 x 180 color display, 75 mW for the pen digitizer, and 1.25 W for the two radios necessary for full-duplex operation. This results in a total power consumption of about 9 W. Thus, the custom chips, which perform most of the computation in the pad, consume a very small fraction of the total power (Fig. 2). In future versions of the pad the programmable logic, microprocessor, and radios will also be replaced by lower-power designs. In addition, recent advances in displays promise to reduce that power component substantially.

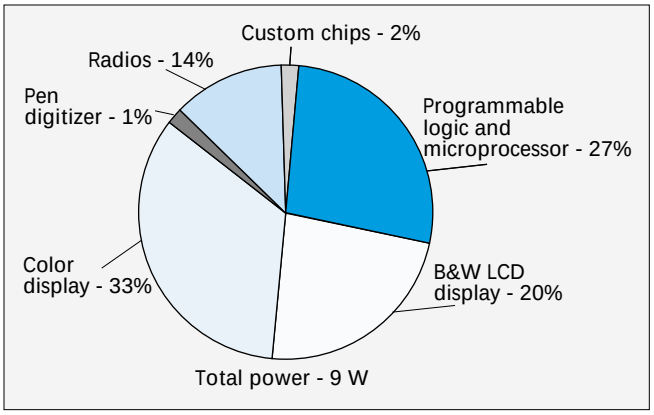
InfoNet

The InfoNet system makes the pad appear to be a fixed-location network-connected terminal. The InfoNet software also provides support for low-latency high-bandwidth communication between the type servers and the pad. To accomplish this, the InfoNet contains a network management and routing system that hides the mobility of the pad and provides communications with performance guarantees.

Design Issues

To meet the above requirements, the InfoNet design requires the software to perform the following functions.

Packet Routing — The InfoNet software routes packets from the type servers to the mobile terminal. Each type server



■ **Figure 2.** *Pad power consumption.*

interfaces to a pad as it would interface to a host on a fixed network. This allows the mobility of the pad to be completely transparent to the type servers and applications. This support implies that the InfoNet must support the handoff of communication between the cells of the wireless network in a quick and efficient manner [5].

Radio Resource Management — The building in which the InfoPad operates is divided into pico-cells with a typical radius of 10 m. The InfoNet software allocates the frequency-hopping sequences and determines when handoff (migration of a user between cells) occurs.

Name Service — All InfoNet processes are locatable via a name server. This server supports frequent updates from the network. It also provides support for non-InfoNet modules, such as applications and type servers, to advertise services.

In addition, we desire the InfoNet to have the following properties:

Support varying qualities of service — Applications supported by InfoPad have widely varying bandwidth and error rate requirements. For example, video requires high bandwidth and tolerates low reliability, while user input tolerates low bandwidth and requires high reliability. To support these requirements, InfoNet should provide the ability to request and guarantee different qualities of service (QoS) on the wired and wireless networks.

Low Latency — Applications and user interface software execute on computers connected to the backbone network. As a result, the observed performance of the terminal is tightly linked to the latency of communication between the compute servers and the mobile terminal. InfoNet provides low latency network communication.

Scalability — InfoNet scales easily to add more users, to use faster communications, and to support additional applications.

It is clear from analyzing the above requirements that system state should be divided into two associations: radio cell and pad. In order to improve efficiency, scalability, and modularity, it was decided to separate the functionality of the control and data paths. This resulted in an InfoNet architecture consisting of three logical modules: gateway, cell server, and pad server. This partitioning of functions is shown in Fig. 3.

For each pad in the system there is a single corresponding pad server running on the backbone network. The pad server is a control entity responsible for managing access to the terminal. It allocates the terminal bandwidth and resources (speaker, microphone, etc.) among the different type servers. To perform its duties, the pad server maintains information about the pad's state, including radio bit error rate (BER), received radio power level, and cell location.

Each radio cell in our system has an associated gateway and cell server. These two modules together provide the functionality of a conventional base station. The gateway performs the data path operations associated with base stations, while the cell server provides the associated control functionality. The gateway routes packets between the wired and wireless networks and performs the protocol conversions necessary between the two networks. The cell server controls the allocation of resources among the pads within the cell. Since resource consumption within a cell often affects nearby cells, each cell server negotiates resource allocations with the neighboring cell servers. Each cell server maintains information about the current status of its cell, such as radio BER and available frequencies.

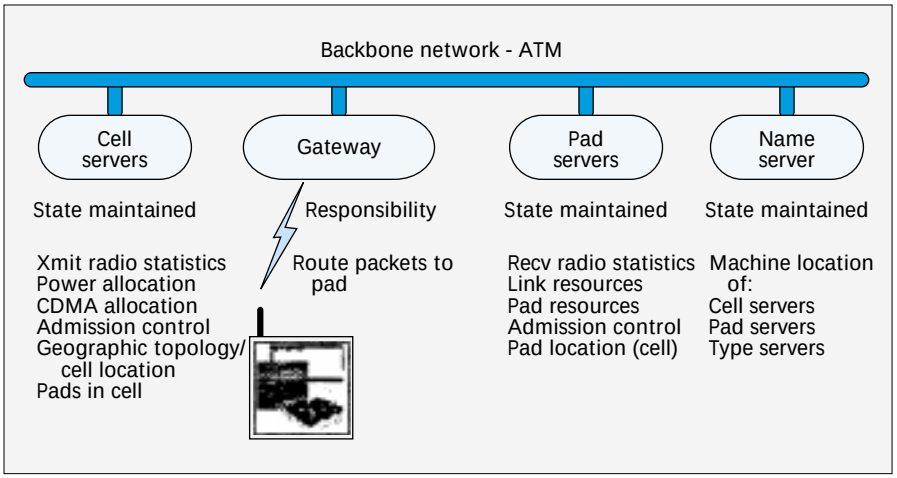
Implementation

A key performance goal of the InfoNet implementation is to minimize latency while providing the full bandwidth of the wireless network to the type servers and applications. The pad server, cell server, gateway, and name server are implemented as UNIX user-level processes. The routing of data between the InfoNet servers and type servers is shown in Fig. 4. Data connections allow the flow of information between the type servers and gateways. Control connections are used to set up and maintain the data connections.

Once a type server is initialized, it begins transmitting data to the pad. InfoNet is responsible for routing these data packets to the gateway associated with the pad. When the pad moves from one cell to another the routing of packets must be updated quickly to avoid data loss. This can be done by updating the routing tables and appropriately forwarding packets that are already in flight. However, this introduces additional latencies that severely degrade the quality of video and audio received at the pad. To perform seamless handoffs, the new gateway must begin receiving packets for the pad prior to the handoff. Multicast provides a natural, low-overhead mechanism to deliver the same packets to both the new and current gateways.

Each type server transmits packets to an Internet Protocol (IP) multicast group [6]. For each pad in the system there is a unique IP multicast address. The gateway responsible for a pad joins the associated multicast channels and forwards any packets it receives across the wireless network. This allows the type servers to transmit packets without knowing the current location of the pad. The pad server keeps track of the pad's cell location in order to configure the routing and perform handoff. It uses the pad's current cell and radio measurements to determine that the pad is nearing a cell boundary and handoff is likely. When initiating a handoff, the pad server requests the gateways of nearby cells to join the corresponding IP multicast group. As a result, all the processing and routing changes necessary for handoff are done before the pad enters the adjacent cell. This enables handoffs to complete quickly and not affect the delivery of multimedia data. This form of handoff has been implemented and measured in a similar system that uses conventional laptops as mobile devices. Handoffs take between 8 and 15 ms to complete, independent of the amount of data in flight [7].

The frequency-hopped radios currently used by InfoPad limit the radio resource allocation needed and the QoS support possible. We statically choose the appropriate hopping sequence for different cells to eliminate interference between users. Therefore, the cell server uses error coding alone to



■ **Figure 3.** *InfoNet partitioning.*

control available bandwidth. A more sophisticated bandwidth allocation algorithm using negotiation between cells awaits the custom InfoPad CDMA radio that supports fine-grained power control.

The above system is stable and is being used by several type servers and applications. The performance of an earlier version of the InfoNet software was measured in detail and is discussed later.

Type Servers

In our system, there is a type server associated with each device available on or emulated by the pad. The InfoNet system presents the type servers with the view of the pad as a stationary, closely connected terminal. The various type servers take this representation and provide the applications with an interface similar to that of a standard workstation. To bridge the gap between these abstractions, the type servers hide the specifics of the pad hardware and provide a device-independent, network-transparent interface for the applications to use.

The type servers follow established application programming interface (API) standards as far as possible, allowing standard applications to operate without modification. For example, a modified X11R6 [8] server provides access to the pad's screen. Similarly, the audio server is built using AudioFile [9], the video server is built on top of the Continuous Media Toolkit [10], and the handwriting recognizer provides an interface based on a Sun Microsystems API [11]. For some of the devices there are no established standards. These servers use custom interfaces that suit our needs. We describe the various type servers in detail in the following sections.

Pen Server

The pen server reads pen packets that arrive from the pad, interprets them, and makes pen data available to all applications. It also emulates the mouse by generating X events and sending them to the X server as mouse events.

We would like to use the X window system to make pen coordinates available to applications. However, X supports only screen pixel resolution coordinates, which have a lower spatial resolution than raw pen data. Pen resolution is crucial for accurate handwriting recognition. Therefore, an alternative mechanism for sending pen-resolution data to applications is used.

As much of the system as possible is independent of the type of digitizing tablet, since future versions of the InfoPad may use a different pen digitizer. Therefore, the pen server normalizes the pen samples to be eight times the screen resolution, independent of the type of digitizer used. Also, the pen server interprets the byte stream from the digitizer and converts it into pen coordinates; it is the only place in the system where this interpretation is done, allowing applications to be independent of the digitizer.

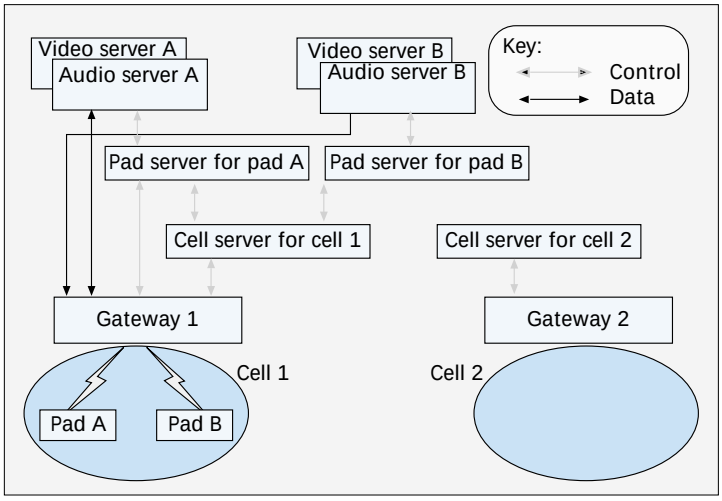
Implementation — The pen server reads a byte stream from the pad and translates it into a series of packets that contain its x and y coordinates, its tip and barrel button status, and its time stamp. Currently, it supports the Logitech PenMan, Wacom SD-510C, Wacom UD-0608, and Scriptel digitizers.

An X event is generated based on pen packets. If the pen server detects that the pen has moved, a Motion event is generated; if a button was pressed or released, a ButtonPress or ButtonRelease event is generated. Pen events occur at 100 points/s, much faster than mouse events on most workstations. This event rate would overwhelm most X servers. Therefore, the pen server subsamples the events it sends to the X server by a factor of 3 in frequency.

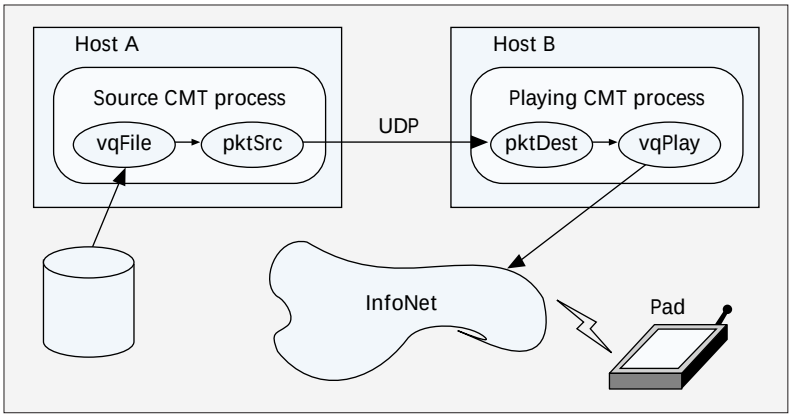
In order to provide useful abstractions to applications using raw pen data, we provide an API that encapsulates a socket interface in an easy-to-use set of functions. Each pen packet is transmitted over the network to all applications that request it. Each application may, at any time, instruct the pen server to stop or resume sending data to it.

Text/Graphics Server

The text/graphics server provides a windowing environment, or graphical user interface, for the InfoPad user. A wide variety of applications are supported under the windowing environment, including simple shell terminals, Web browsers, text editors, and animation/video applications. We modified an X11 server to provide text/graphics support. Several features of X are attractive, including its distributed client/server model, its platform independence, its ubiquitous use in the



■ **Figure 4.** *InfoNet data routing. Cell 1 has two pads, A & B, in its coverage area, whereas cell 2 has none. Each pad has a video and audio type server running. Arrows depict routing of data.*



■ **Figure 5.** *Example of CMT video delivery. The video is stored at host A. Host B, which is near the InfoPad on the network, synchronizes the video with the audio.*

UNIX workstation environment, and its ready source code accessibility.

The text/graphics server formats the drawing data into a form understandable by the pad's frame buffer. In addition, the text/graphics receives pointer events from the pad via the pen server.

When InfoPad users move a window, pull down a menu, scroll within a window, or draw with the pen, they expect "good performance." Response time to an input action must be sufficiently rapid that the user can interact with the applications naturally. The user also expects a certain quality in the screen presentation; that is, text should be legible and images recognizable. However, the wireless link has limited bandwidth and is prone to data loss, resulting in poor performance. Thus, an important function of the text/graphics server is to compensate for the imperfections of the channel by some form of source and channel coding.

Implementation — Since the frame buffer is remotely located on the pad, the X server needs to be split at some level. We split it at the lowest level of monochrome bitmaps, instead of sending command-based primitives to be rendered on the pad. Downloading bitmaps has several advantages: error resiliency, independence of pad design from text/graphics server choice, and the flexibility to employ image processing and coding techniques. There are several methods to improve performance in the split-X architecture. First, grouping of pixels and lines of pixels into rectangular blocks greatly reduces latency. Second, partitioning the split-X server into two processes with shared memory between them significantly enhances pen inking performance. One process functions as a conventional X server, while the other transmits bitmaps to the pad.

As part of our efforts to deal with errors introduced by the wireless channel, we display all bitmaps even if they are corrupt. During interactive operations such as menu pulldown, it is better to display a possibly corrupt initial version quickly than to wait for a completely correct but delayed version. Later, when the user is inactive in a certain region, a background refresh algorithm cleans up any artifacts.

Audio Server

The audio server provides applications with an interface to the audio input and output ports. Applications can read and write the audio port regardless of the presence of other applications using the same port concurrently. In order to adhere to current standards, we modified an AudioFile server to work on our system. AudioFile meets our requirement of concurrent access from several clients and is used by several existing applications.

Implementation — The audio server reads a byte stream from the pad and stores it in a circular buffer. An application may connect using the AudioFile API and read data from this buffer. The buffer has a limited length, so it must be read frequently to avoid data loss. The audio server maintains a concept of absolute time, and the API allows the application to control how many seconds or milliseconds of data to read and the absolute time index of this data.

The application can send audio data to the pad through the audio server. Since there is little audio buffering on the pad, the audio server delivers sound samples to it at a controlled rate to avoid buffer overflow. The downlink functions similarly to the uplink in terms of time synchronization: the circular buffer is indexed by time, and applications have to specify at what time the data should be played. If multiple applications write to the audio port at the same time, the audio server mixes all the inputs into the downlink buffer. If there are no currently writing applications, the audio server detects silence in the downlink and stops sending data. This helps reduce the network and radio bandwidth used by the terminal.

Video Server

The video server is responsible for delivering synchronized video and audio from arbitrary sites on the wired network to the pad in real time. We ported the Continuous Media Toolkit (CMT) developed by the Berkeley Plateau Group to the InfoPad system. The CMT was modified to use VQ, the InfoPad video-encoding format.

Network bandwidth is an issue in all video applications. In general, the wired network will be able to deliver much more bandwidth than the wireless link can handle. We want the system to use only as much bandwidth as necessary on the wired network rather than transmitting frames that are never displayed.

Implementation — Two modules were added to the object-oriented CMT. The first, `vqFile`, deals with reading VQ frames from disk; the other, `vqPlay`, formats the data appropriately, performs final synchronization with audio, and sends the data to the pad. Since the CMT supports the same AudioFile interface that is provided by the audio server, no modifications were necessary to support audio synchronization.

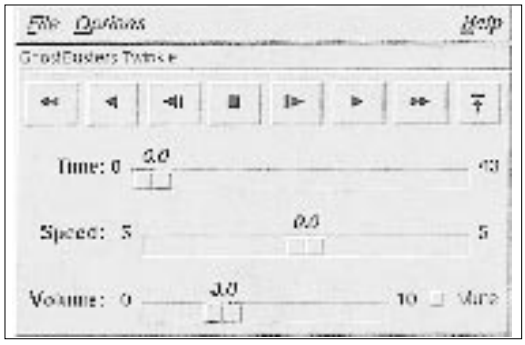
Running a CMT server at the media sources and a CMT player near the pad meets most of the video requirements of the InfoPad system. The CMT has several standard objects that make network communication simple, such as a pair of objects for sending data from one process to another using best-effort UDP (`pktSrc` and `pktDest`). An example CMT configuration for playing video from a remote host is shown in Fig. 5. Audio is handled similarly using a separate stream.

All parts of the CMT share a common logical time system, enabling them to synchronize audio and video, even from disparate sources. It is also easy to change the speed at which logical time changes relative to physical time and to set the logical time instantaneously. This allows us full VCR-like control over stored video (Fig. 6), including playing at any speed, forwards or backwards, as well as features not found on VCRs, such as random access.

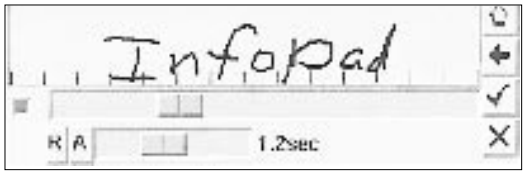
Handwriting Recognizer

The responsibility of the handwriting recognizer is to allow applications to use the pen device to obtain textual input. Since the pad is primarily an information retrieval and com-

munications device, we believe it is unlikely that the pad will be used to write large amounts of text such as computer programs or books. Rather, users will control multimedia applications, send and receive electronic mail, and perform other similar activities. For controlling applications, they will enter names of files, World Wide Web (WWW) URLs, user names, and other words that may not be found in a dictionary. Unfortunately, cursive recognition requires a dictionary in order to obtain reasonable accuracy. Furthermore, with the advent of Multipurpose Internet Mail Extension (MIME)-capable mail readers, electronic mail may not necessarily be in ASCII but in forms such as electronic ink [12]. Recognition is therefore not strictly necessary for e-mail, although a cursive recognizer would ease the use of ASCII e-mail. Based on these requirements, the handwriting recognizer supports the recognition of printed rather than cursive or mixed-mode handwriting.



■ **Figure 6.** *InfoPad video controls.*



■ **Figure 7.** *Handwriting recognition widget.*

Since the handwriting recognizer uses compute-intensive algorithms, the computation is performed on a remote processor. Programmers are provided with abstractions to access the recognizer and need not know it is running remotely nor that text is being entered via handwriting recognition. Two interfaces to the handwriting recognizer are provided: a full-featured one used by applications that wish to handle the digital ink themselves, and a simple one that abstracts the entire handwriting interface.

Implementation — The handwriting recognizer supports only printed text. The vocabulary is predefined to be the characters A–Z, a–z and some punctuation characters. There is a separate recognizer that handles digits 0–9. Both recognizers use the same hidden Markov model engine with different vocabulary files.

The recognizer was trained on a small set of data captured from several writers. The practical difficulty of obtaining a large training set led to a compromise: we focused on the user interface and made the recognizer a plug-and-play module that can be replaced when better commercial or research recognizers become available. The current recognizer uses very simple features based on slope, difference-of-slope, and y-coordinate sequence. For characters that have identical shapes in both upper-case and lower-case forms, writing area size information must be passed to the recognizer.

As mentioned above, two different interfaces to the recognizer are available. The more sophisticated interface is based on the Sun Microsystems API standard for handwriting recognition engines. The recognizer engine is compiled into a shared object library which is dynamically loaded at runtime. This API allows control information to be fed to the recognizer and allows the recognizer to return the top few candidates, from which the application may use special knowledge to select. This API is used by the handwriting recognition widget (Fig. 7) described below. Alternatively, a simple socket interface (with API) is available by which the recognizer reads directly from the pen server and sends recognized text to the application. Using this API, the application tells the recognizer which areas of the screen to monitor.

To facilitate investigation of the role of handwriting in user interfaces, we have extended the Tk [13] GUI toolkit with a pen input widget that transparently handles stroke data collection, digital ink display, cutting and pasting of stroke data, and interaction with one or more recognizer engines. We have also exported most of the Sun Microsystems handwriting recognition API to the Tcl interpreter, so user interface experiments such as arbitration among multiple recognizers, implementation of various error correction models, and dynamic changes to the supported vocabulary can be performed entirely in Tcl. The widget's modular input/output (I/O) architecture allows stroke data to be received from a mouse, tablet device, or arbitrary event-based stream. In its current implementation, the widget is a standalone application that displays a writing window. Characters written in this window are recognized and sent, via the X window system, to the most recently used Tk entry widget. The widget is being modified to function as a drop-in replacement for the standard keyboard text entry widget.

Speech Recognizer

The objective of the speech recognizer is to provide InfoPad applications with a mechanism to receive commands and simple data by recognizing user speech. For example, the recognizer allows commands that are usually issued by selecting menu items, clicking buttons, or pressing function keys to be executed by speaking to the pad. Just as the names and contents of menus and button bars change from application to application, so do the vocabulary (list of pronunciations of words that can be recognized) and grammar (the order in which these words may be recognized). The speech recognizer accepts continuous speech (no pauses are required between words) and is speaker-independent, but is capable of some adaptation to individual speakers. In its current form, it is intended for small to medium-sized vocabulary applications such as command and control operation, but not general speech-to-text transcription.

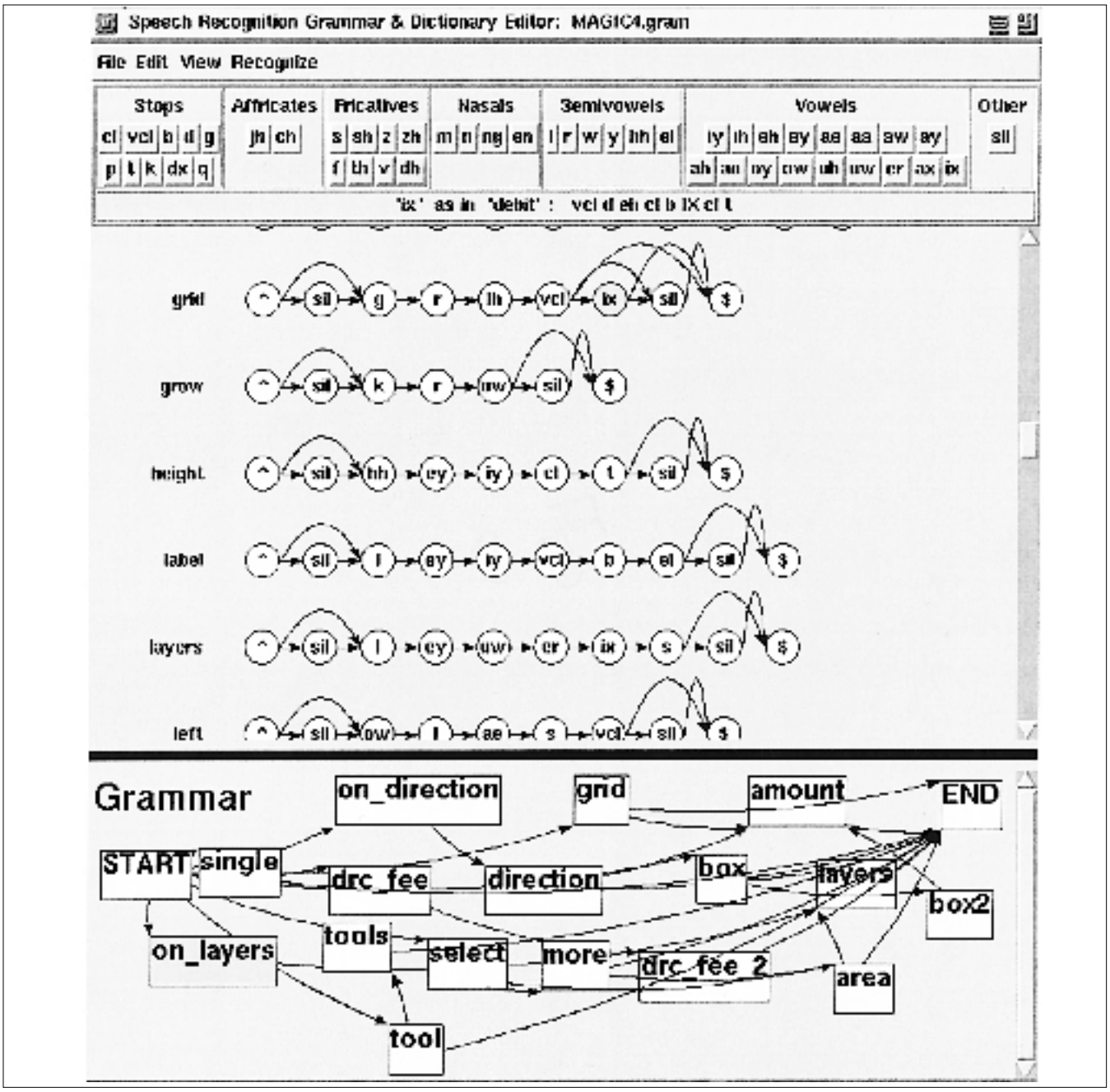
The speech recognizer was designed as a compromise between the need to couple the recognizer tightly to the individual applications in order to increase recognition accuracy and the desire to make it as simple as possible for programmers to use speech recognition. Unfortunately, it is difficult to design a speech recognizer that can accurately recognize arbitrary English sentences, especially if speech is continuous. However, we can often narrow the recognition to a particular subject, such as commands one might give to a Web browser. This allows the recognizer to greatly reduce the size of the vocabulary and the complexity of the grammar that it must consider, thereby enhancing accuracy. Thus, it is important for the applications to tell the speech recognizer in advance the vocabulary and grammar that it expects to receive from the user. This is a fairly simple task if the program is receiving verbal command and control: programs already have vocabularies and grammars defined by their button names, menu hierarchies, and typed commands.

Implementation — Programmers know what commands their programs can receive; the challenge is to make it easy for

them to pass this information to and from the speech recognizer without forcing them to understand the details of speech recognition. We meet this challenge by implementing the speech recognizer as an extension to the Tcl language, by providing high-level Tcl/Tk widgets and itcl objects for applications to interface with the recognizer, and by providing a graphical vocabulary and grammar editor.

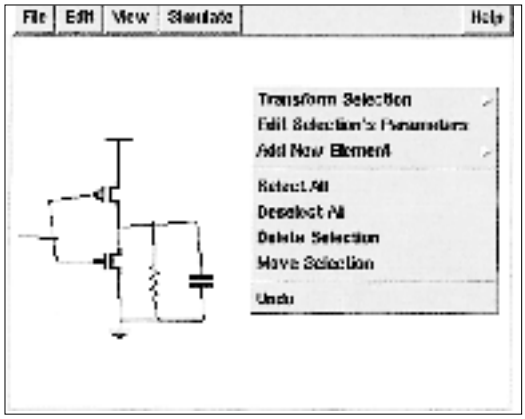
All interaction with the speech recognizer can be handled through two Tcl commands, one controlling the recording of sentences and the other the actual recognition. Since the continuous-word recognizer recognizes entire sentences, not just words, the recorder supports automatic silence detection to locate the beginning and end of sentences. To facilitate error recovery, the recognizer provides a list of several top matches; thus, if the best estimate of the spoken sentence was not correct, the user can make a correction from the other choices. The recognizer is capable of determining if the user speaks a word that is not in its vocabulary rather than simply making the closest, albeit poor, match.

While the Tcl commands allow complete, low-level control over recording and recognizing speech, the most common tasks are controlled by the programmer through high-level itcl objects and by the user through itcl mega-widgets. These objects contain procedures to allow the programmer to manipulate vocabularies and grammars and to perform recognition without having to deal with the underlying data structures. These procedures also enable the programmer to convert spoken words into their phonetic transcriptions so that new words can be added to the vocabulary as they are spoken by users. A similar mechanism allows users to train



■ **Figure 8.** *GramCracker application for creating and modifying speech recognition vocabularies.*

the recognizer to adapt to their own particular pronunciations of any given word if it differs significantly from the "standard" speaker-independent pronunciation. The mega-widgets are designed to be dropped into programs to give users access to all necessary controls (e.g., volume control and silence detection levels), speaker adaptation, and recognition results. Finally, the programmer can create and edit vocabularies and grammars using the GramCracker application (Fig. 8), which allows graphical display of pronunciations and grammars.



■ **Figure 9.** *Circuit schematic entry application, with Edit menu visible.*

Applications

The type servers provide compatibility with standard workstation applications, allowing many off-the-shelf software packages to work unmodified on the InfoPad system. However, many of these applications are not well suited to the InfoPad environment. They may not operate well with the small screen size or take advantage of the characteristics of handwritten or spoken input. We built several applications to demonstrate the viability of our system. Some applications show methods to overcome the obstacles of portable operation, while others introduce new functionality made possible by our system.

Each application showcases the capabilities of one or more components of InfoPad. The circuit schematic recognizer demonstrates the use of pen input and handwriting recognition, while the Web browser highlights the use of speech recognition. The Multicast Backbone (MBone) [14] tools were ported to the InfoPad to demonstrate our conferencing capabilities. The unique aspects of these programs are described in the following sections.

Circuit Schematic Recognizer

The circuit schematic recognizer allows users to draw simple electrical circuits on screen and generate Simulation Program with Integrated Circuit Emphasis (SPICE) [15] simulation decks. This application allows us to explore the use of the pen as a design tool and the synergy between audio and pen input in an application. It uses the pen as a drawing tool and for accessing menu-driven commands, the handwriting recognizer for entering parameters and names, and the speech recognizer for commands. It recognizes drawn circuit elements and automatically decides whether drawn items are elements or wires.

Using pen and audio input in this application allows the user to overcome several limitations inherent in current desktop circuit schematic entry systems. Keyboard-mouse systems require the user to alternate between keyboard and mouse, navigate between the menu bar and the drawing canvas, and manually rotate and reflect objects after they are placed. This application changes the way users perform these operations and makes these operations more convenient by taking advantage of the unique features of pen and audio input.

Several principles and constraints influenced the design of this application. Although the pen has two buttons, we found that users often press the barrel button by accident and, when asked to press the barrel button on purpose, often have difficulty locating and pressing it. Therefore, this application does not rely on the barrel button. Users often have difficulty with double-tapping a pen because they have to lift a pen to do so; the second tap often comes down far from the first tap, so spatial accuracy is lost. In addition, the pen tends to bounce, causing multiple taps to be recorded. Therefore, we use only single pen taps. We also found that speech recognition was difficult in noisy environments where this application may be used. As a result of these concerns, the circuit recognizer allows all functions to be accessible using the pen interface. However, speech recognition is supported to improve usability in quiet environments.

The circuit schematic recognizer starts up by presenting a blank drawing canvas and a menu bar along the top (Fig. 9). The primary method for specifying circuit elements and wires is by drawing. Users may draw circuit elements or wires where they would like the items to appear. The recognition algorithm then compares the drawn sequence of strokes to templates of circuit elements to find a match. This comparison uses a priority ordering of most complex element first. Currently, this application supports negative-channel and positive-channel metal oxide semiconductor (NMOS and PMOS) transistors, resistors, capacitors, power and ground circuit elements. It also recognizes rotations and reflections of these elements. If the drawn item is not identified as a circuit element, the application assumes it is a wire. Alternatively, circuit elements may be specified via the edit menu (and therefore by voice commands); this menu may be brought up by tapping the pen on open space.

Once a stroke sequence is interpreted as a circuit element, the application replaces the strokes with a bitmap of the element. The recognizer rotates and reflects the bitmap to match the orientation of the drawing. The program only supports rotations in multiples of 90 degrees and reflections about the x and y axes. The program aligns elements to connect to adjacent wires if the stroke endpoints are close enough to the wires in question.

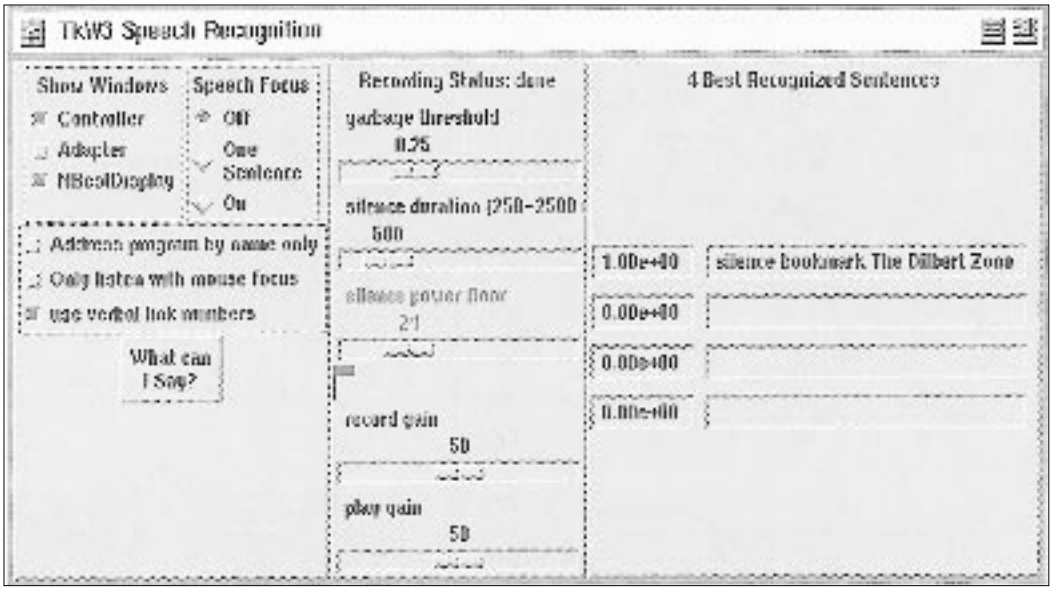
If a stroke sequence is interpreted as a wire or series of wires, the strokes are erased and replaced by straight-line segments. The recognizer aligns these segments to connect to adjacent segments or circuit elements.

The user must specify the parameters (e.g., resistance) of many circuit elements. Clicking on the appropriate button in the edit menu presents the user with a form in which labels and relevant parameters for an item or wire are entered. The parameter entry form uses Tk entry boxes; the user writes into the handwriting recognition widget (see "Handwriting Recognizer"), which automatically processes the electronic ink and prints the recognized text in the entry box.

World Wide Web Browser

The WWW browser demonstrates InfoPad's text, graphics, audio, and video output along with its pen input and speech and handwriting recognition abilities. The browser is based on tkWWW [16], a Tcl/Tk-based WWW browser available under GNU Public License. We modified the program to add several features, most notably in-line images and speech recognition control (Fig. 10).

The WWW browser incorporates speech commands at three levels of sophistication. The first and simplest form implements button and menu commands such as "reload," "back," and "go home." These commands are always present and use speaker-independent pronunciations, although the



■ **Figure 10.** *Speech recognition control window for tkWWW.*

pronunciation can be customized by the user to enhance accuracy.

The second form of speech command is used to implement verbal bookmarks. Since bookmarks are added and deleted by the user, the parts of the vocabulary and grammar that pertain to bookmarks are changed dynamically. It is unreasonable to expect the program to guess the pronunciation of page titles such as "InfoPad Applications/UI Subgroup." Instead, the application records the user saying the name or nickname for the page ("apps group"), transcribes the name, and thereafter uses that pronunciation as the verbal bookmark (i.e., saying "bookmark apps group" will jump to that page).

The third form enables speech activation of embedded hyperlinks (as a supplement to tapping the pen on the links). Determining the pronunciation of a link's name is a comparable problem to determining the pronunciation of a bookmark name; the problem is exacerbated by the tendency of HTML writers to name all of their links "here" (as in "click here"). InfoPad's Web browser uses two approaches to verbally identify links. First, it allows the HTML page writer to specify the pronunciation of the link, as well as an audio file containing someone speaking the name of the link so the user can listen to the names of all available links. This additional information is embedded as attributes in the link, so no nonstandard extensions to HTML are required. For those links that do not have verbal information built in, the browser can display a unique number next to each link on a page; the user merely speaks "jump three" to activate the third link on the page.

The browser uses handwriting recognition to allow users to access WWW pages by their URLs. When the user says "goto," she/he may write the URL in the pen entry widget.

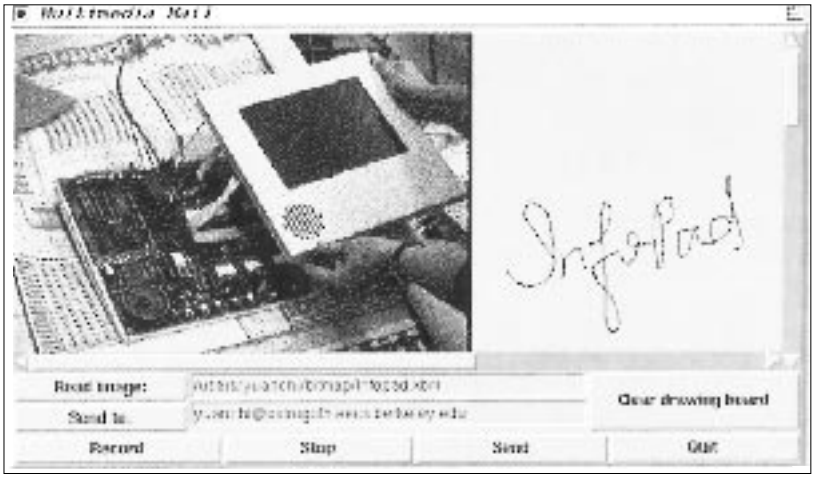
Multimedia Mail

Multimedia mail is a MIME-compliant mail sender. It allows users to compose messages containing audio, images, and hand drawings, and send them via electronic mail (Fig. 11). The received multimedia mail message may be viewed by any properly configured MIME-compliant mail handler.

This application enables a user to send out e-mail messages that fully utilize the multimedia features of the pad. It uses the services of several type servers: the audio server for voice recording, the pen server for hand drawing, and the handwriting recognition server for specifying command parameters.

The hand drawings are sent as pen strokes rather than bitmaps to reduce the size of the message. The size of an uncompressed audio and image message can easily reach several hundred kilobytes, so future mail programs of this kind will need to use data compression.

Careful attention was given to the design of the application's graphical user interface. Since the display resolution of the pad is relatively low compared with that of a workstation, the application is designed to fit in the screen, and scroll bars are used on the image and hand-drawing canvases. Although handwriting recognition is used by this application, the user interface is designed to reduce the need for command parameter input. Frequently used parameters, such as e-mail addresses and image file names, are built into pulldown menus.



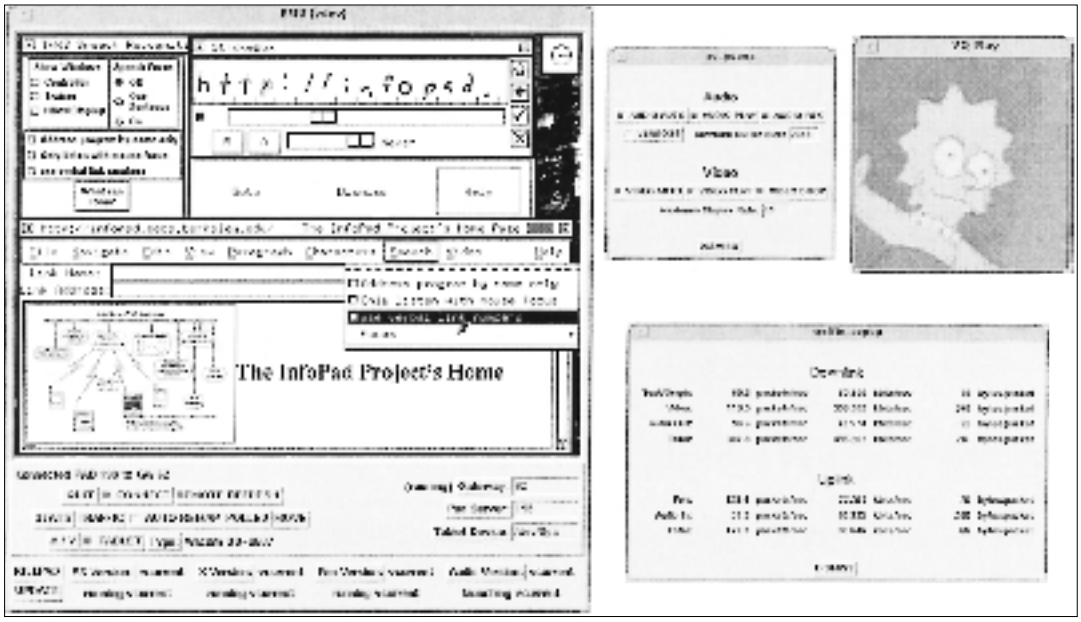
■ **Figure 11.** *A multimedia mail message.*

MBone Tools

One of the important target applications for the InfoPad system is the use of the pad as a conferencing tool. Currently, the most popular conferencing tools on the Internet are the MBone utilities: `vat` (Visual Audio Tool), `wb` (White Board), and `vic` (Video Conference).[1] We felt it was important to provide special support for the use of these tools on the pad. `Vat` and `wb` run on our system as standard X11 and AudioFile applications. Unfortunately, `vic` performs very poorly as a standard X11 application on the pad. `Vic` normally uses the X11 shared memory exten-

[1] *These tools are available from URL ftp://ftp.ee.lbl.gov/conferencing*

sion to provide reasonable video playback performance. This extension is unavailable on the InfoPad system because all graphics data must be transferred across the network before display. To run `vic` efficiently, we decided to transcode the video display data to the InfoPad VQ format. A video gateway (`vgw`) [17] that supports conversions in real time between H261, Motion Picture-Joint Photographic Experts Group (MJPEG), and the InfoPad VQ formats is used to perform the trans-coding. The `vic` tool also required minor modifications to use this video gateway. The combination of the modified `vic` and `vgw` provides video conferencing at a resolution of 160 x 120 and a rate of 30 frames/s.



■ **Figure 12.** *The InfoPad emulator.*

Performance

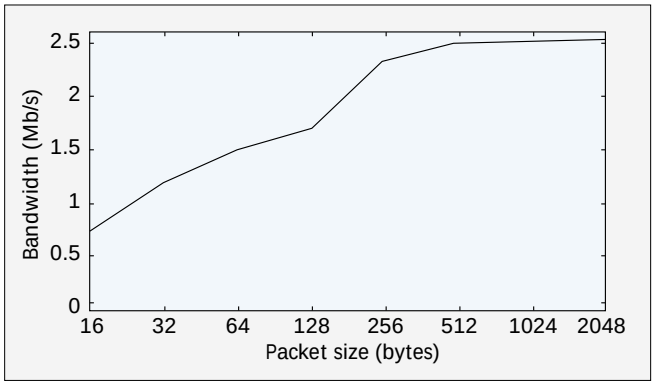
The pad hardware supports few debugging and profiling aids because it was kept simple in order to reduce power consumption and weight. A software-based InfoPad emulator (Fig. 12) was used to test and measure the performance of our system. The emulator is an X application that runs on a standard workstation without the use of any custom hardware. It provides the full user input and output capabilities of the actual pad hardware, including text/graphics, full-motion VQ video, and audio output, as well as pen (or mouse) and audio input. The debugging and profiling aids it provides include detailed data rate monitoring, server status display, and emulation of the actual pad's hardware capabilities.

The emulator receives and transmits the same data packets that the hardware pad does. In the case of the emulator, the data transmission is performed over a Transmission Control Protocol (TCP)/IP connection between it and a gateway. This is in contrast with the case of the hardware terminal in which the gateway communicates with the pad over a radio link. However, in both cases the full infrastructure of type servers and InfoNet servers is identical.

Pen loop-back latency is used as a metric of the overall performance of the software system. Loop-back latency in this context is the time between pen contact with the pad and the appearance of ink on the screen at the pen tip. This exercises the data path through the InfoNet system, the pen server, and the text/graphics server. A set of performance measurements conducted on an earlier version of the system indicated that pen loop-back latency could be reduced to about 10 ms. InfoPad has set a target ceiling of 30 ms for this latency. Current prototypes exhibit delays somewhat longer than 10 ms due to additional complexity, but pen latency is still visually excellent.

Network Measurements

An early prototype of the InfoNet architecture was built to determine the feasibility of the system — in particular, whether it could meet the latency requirements of applications and support the bandwidth available on the wireless network. There are several significant differences between this prototype and the current InfoNet implementation. In the prototype and current implementation, the pad server, cell server, and gateway are implemented as UNIX user-level processes and communicate using TCP/IP. In the prototype the type servers transmit their data to a pad server using TCP connections, whereas in the current implementation data is transmitted directly to a gateway using IP multicast channels. Each pad server maintains a TCP connection to the gateway forwarding packets to the pad. In the prototype, the pad server uses this connection to route packets between the type servers and gateway.



■ **Figure 13.** *Peak bandwidth between the type servers and the pad.*

Performance measurements of the InfoNet system focused on throughput, packet rate, one-way delay, and delay jitter at various packet rates and sizes. These measurements were obtained using a special benchmark type server, the early prototype of the InfoNet software, and a 10BaseT unswitched Ethernet. The three major findings are:

- The limiting factor for packet rate is overhead from kernel crossings. The frequency of read and write system calls is limited primarily by central processing unit (CPU) speed. At packet sizes above 256 bytes, bandwidth limits the packet rate.
- The peak bandwidth available between a type server and

pad is shown in Fig. 13. The network capacity of the Ethernet used limits the peak bandwidth possible. During these measurements, each packet from a type server passed through a pad server and gateway before being delivered to a pad emulator. As a result, each packet traversed the network three times resulting in the peak observed bandwidth of 2.5 Mb/s over Ethernet. Higher-bandwidth networks, such as ATM or 100BaseT switched Ethernet, should alleviate this bottleneck somewhat. For packets sizes between 256 bytes and 1 kbyte, we believe this InfoNet prototype will support about 3–5 Mb/s. In addition, the current InfoNet implementation with multicast routing performs significantly better.

- The system under test is remarkably well behaved with respect to end-to-end jitter and delay below specific transmission rates.

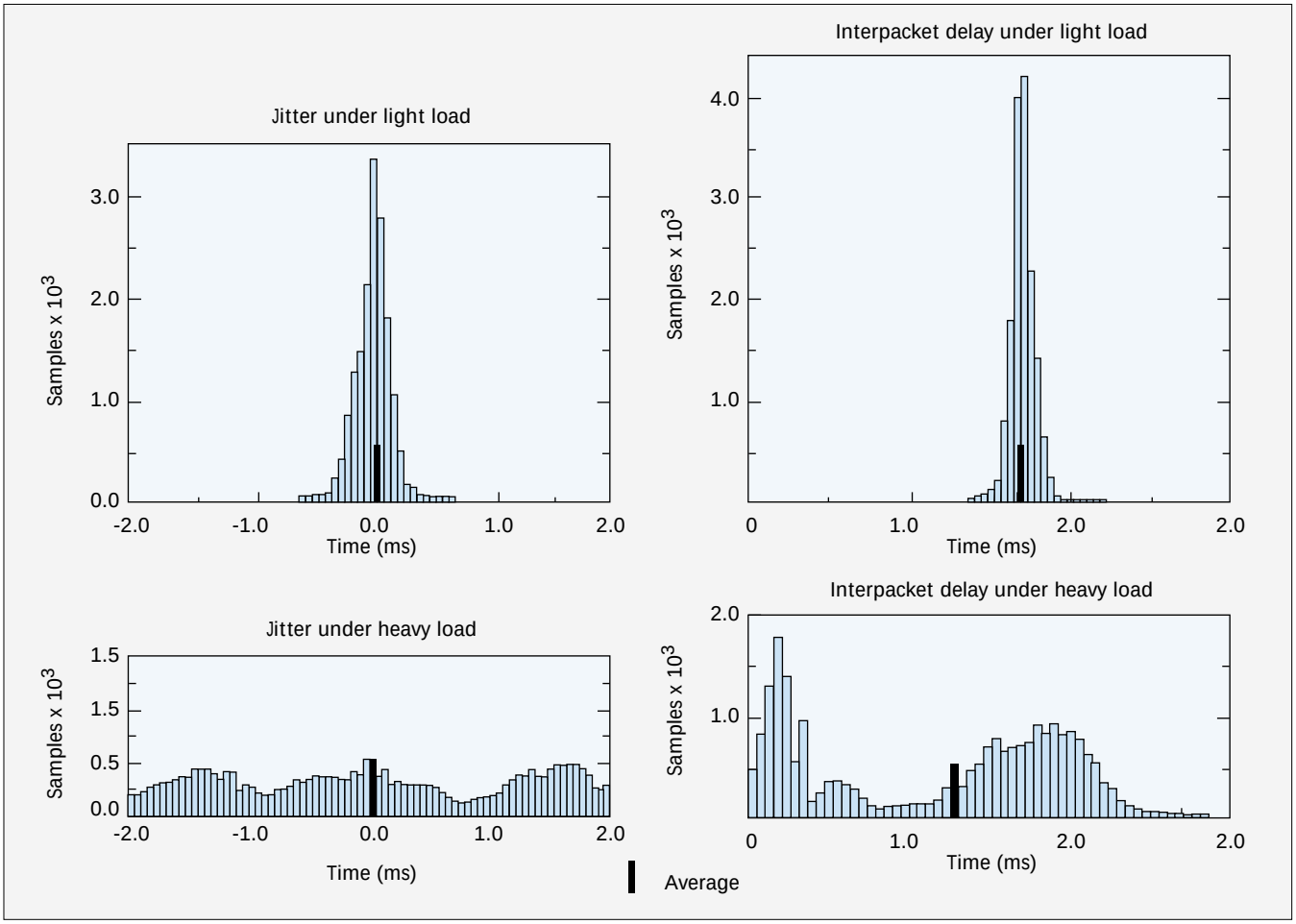
Interpacket delay and jitter are best represented by their distributions [Fig. 14]. At low packet rates (less than 700 packets/s for 256-byte packets) both have a single, very sharp mode, interpacket delays around 2 ms, and jitter around 0 ms. When the packet rate reaches a certain point interpacket delay begins to show signs of a bimodal distribution, with one mode at 2 ms and the other near zero. Similarly, jitter begins to show signs of a trimodal distribution, with the center node at 0 ms and the others symmetrically placed about the center. At higher packet rates, the upper delay mode is reduced and the lower increases, while jitter tends toward a uniform distribution. These phenomena are results of buffering in the system. Although performance may be adequate under some high-packet-rate conditions, the point of degradation to multimodal distributions is considered the performance limit of the InfoNet system. For the InfoNet prototype this limit occurs around 1.5 Mb/s for 256-byte packets.

Scalability

An important performance goal of the InfoPad system is to support as high a user density as possible. The future InfoPad radio supports a maximum of 50 channels (and therefore users) per cell. The network and application software aim to support at least this density. In our system, possible bottlenecks arise from the placement of applications and type servers on compute servers, the routing of packets to the mobile host, and the bandwidth usage on the fixed network.

Since all applications and type servers run on the backbone network, the system needs low latency and high bandwidth between a pad and its associated compute servers. If the latency grows or the bandwidth drops, the interactive performance of the pad degrades rapidly. Our experience with the system indicates that latencies up to 30 ms are easily tolerable. Since each pad is given a 2 Mb/s channel on the wireless network, this bandwidth should also be available between the type servers and gateway. These requirements prevent the pad from operating in environments where high-speed wireless networks are unavailable.

The routing of packets from the type servers and applications determine the latency and bandwidth to the pad. As



■ **Figure 14.** *Example jitter and interpacket delay histograms.*

mentioned previously, the InfoNet system uses IP multicast to route packets to the gateway. The IP multicast routing algorithms are known to work extremely well in networks with a small number of router hops. Therefore, the routing of packets to the gateway does not present a significant bottleneck over normal IP routing. This implies that the type servers and applications may be a few network hops away from the mobile host and still provide reasonable performance.

To complete the delivery of packets, a single gateway has to be capable of routing packets destined for all 50 users in its cell, a load of 100 Mb/s under full utilization with future CDMA radios. The current gateway is implemented as a user-level process on off-the-shelf network and workstation hardware. This implementation is capable of delivering between 3 to 5 Mb/s to a wireless cell, depending on the traffic characteristics. By moving the routing code into the operating system kernel, we estimate that approximately 30–40 Mb/s of traffic can be supported. This may be sufficient bandwidth for 50 users, assuming a reasonable amount of statistical multiplexing. We are currently examining mechanisms of connecting a single cell's radio system to multiple workstations. This would allow us to distribute the gateway processing across these workstations, removing this bandwidth bottleneck.

Multiple cells in an area may consume up to 100 Mb/s each, placing a significant load on the backbone network. To prevent hot spots in our network, we have chosen to use a switched network technology such as ATM or switched 100BaseT Ethernet. We also require that the underlying network technology have efficient support for multicast. The current implementation uses 10BaseT Ethernet.

Our current implementation has some difficulties scaling to the desired user densities. However, with some minor modifications, such as a move to in-kernel routing and a change of backbone network, we hope to support 50 users and 100 Mb/s within a single room.

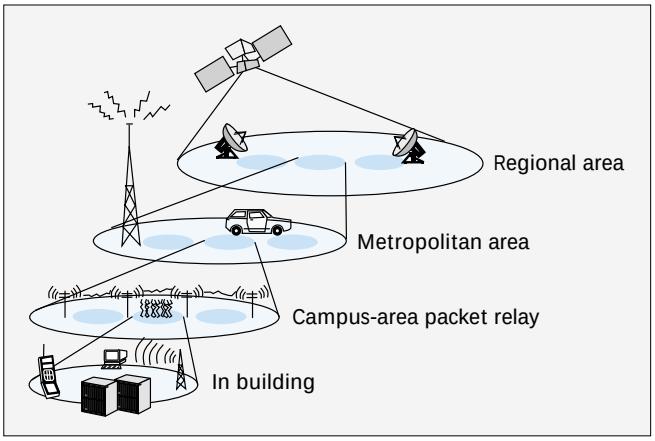
Future Directions

There are three significant efforts being made within the University of California at Berkeley to extend the InfoPad system. There is continuing work on improving the InfoPad infrastructure through the addition of new hardware and software. The Daedalus project is attempting to apply some of the lessons of the InfoPad system to build a system using more general-purpose, programmable, commercially available laptops and PDAs. The third ongoing project is the Medley project, which is developing multimedia transport and coding algorithms more appropriate for wireless systems. These three efforts are described in the following sections.

InfoPad

The InfoPad group is continuing to enhance its mobile computing environment. These efforts range across the hardware and software of the system.

The bus architecture of the pad allows new hardware modules to be added easily. We plan to incorporate new chips that implement different algorithms for receiving and transmitting data. For example, the vector quantization algorithm currently being used for video compression will be replaced by a more sophisticated subband coding scheme. This subband-coded video provides better picture quality and is better suited to the characteristics of wireless communication. We also plan to use the InfoPad to test the efficiency and effectiveness of alternative wireless technologies, including spread spectrum, TDMA, and infrared communications. In addition, we are investigating various error correction strategies and protocols for the wireless link.



■ **Figure 15.** *Wide-area overlay networks.*

The software of the system will be modified to explore network QoS guarantees. The group is examining the use of transmission power, bit error rate, and error correction coding to support the transmission of data types with widely varying requirements for bandwidth, latency, and error tolerance. We plan to allow type servers to request bandwidth and device resources from the pad server over a TCP control connection. If the request cannot be met, the pad server will ask the cell server to allocate more of the cell's resources to the associated pad. Since the raw bandwidth to a pad is fixed by the wireless links, the cell server will perform bandwidth allocation via a combination of interference control and error correction coding. The interference will be controlled by regulating the transmission power or frequency-hopping sequences of the radio system. Reducing interference will lower the error coding requirements of a channel and thereby increase the available bandwidth.

Other plans for the future include improvements to the quality of the video display through the use of a color liquid crystal display (LCD) panel. To support this evolution, the trade-off between putting higher-level drawing primitives in the pad and continuing to transmit raw bitmaps to the pad will be investigated. We also plan to explore the use of specialized computer servers. Computation-intensive modules such as recognizers and video transcoders will be modified to take advantage of parallel computation and accelerator hardware.

In the long term, the InfoPad system must be useful in more than the indoor pico-cellular environment currently targeted. Making this possible will involve the implementation of a multimodal radio that can communicate via lower-bandwidth wide-area radio services. This more flexible InfoPad will have increased requirements for general-purpose embedded processing. We will explore modifications to the software architecture needed to support this.

Daedalus

The InfoPad model works well in an indoor environment where significant computational and networking infrastructure are available. The Daedalus group is examining the difficulties of having a portable device capable of operating in both indoor and outdoor environments and adapting to the capabilities of the available infrastructure.

Wireless networks of the future are likely to be heterogeneous where each host will be simultaneously connected to different wireless network interfaces. Figure 15 shows a set of different overlay networks that a single portable may be capable of accessing. These *overlay* networks are characterized by different coverage, bandwidth, latency, and bit error rates. The Daedalus group is developing network management soft-

ware and application support services to allow mobile computers to dynamically adapt to the state of their network connectivity. The problems being examined include improving TCP performance [18], supporting routing and handoff in heterogeneous networks, characterizing network performance, and supporting applications that are aware of the quality of their network connections and adapt to changes in this quality [19]. The group is also developing applications to drive the design and validate the interfaces between applications and the network. The primary goal of the project is to allow users to access the same applications and information despite significant changes in the quality and state of their network connectivity.

Medley Graphics and Video Transport

Two important performance objectives in the InfoPad system are minimizing the round-trip delay from terminal to type servers and applications, and the interrelated problem of maximizing the traffic capacity of the wireless link. Both these problems are most successfully addressed by coordinating the design of the media coding and the wireless transport system. MPEG is deficient in a fading and interference-limited wireless environment because it minimizes the bit rate without regard for the traffic penalty of the high reliability requirements imposed thereby [20]. We are therefore designing and implementing in future InfoPad prototypes coordinated video and graphics coding and transport systems that introduce a number of innovations beyond past efforts. At the same time, we are carefully addressing many issues inherent in future heterogeneous transport and terminal environments that combine wireless access to a broadband backbone, of which InfoPad is an early example.

To ensure good system modularity, minimize delay, and admit privacy by end-to-end encryption, the InfoPad avoids video and graphics transcoding within the transport system (and the base station in particular), in contrast to existing wireless telephony systems. We enable joint source/channel coding by provisioning *substreams* or *flows* transparently from type server to terminal (similar to a recent Internet protocol proposal) and configuring the QoS parameters (loss, corruption, and delay) for each substream, thereby enabling the wireless transport to allocate resources appropriately for each packet. In the InfoPad CDMA transport, corruption will be controlled by downlink power control and intercell power allocation [21], and delay and loss will be controlled using earliest-deadline scheduling of packets in the base station.

In a coordinated innovation, we have proposed [22, 23] and will implement a new approach to *asynchronous video* coding that achieves joint source/channel coding in the delay as well as loss and corruption dimensions (by abandoning the conventional synchronous model of frame-by-frame video reconstruction). This allows the perceptual delay of the video to be less than the worst-case network delay, and the traffic capacity to be increased by relaxing the worst-case delay and increasing traffic smoothing. Motion-adaptive corruption and delay QoS minimize the perceptual impairment in a heavily loaded cell.

Similarly, our approach to graphics coding will involve a multiple substream model of coordinated source and channel coding for improved response and better image presentation. The multiple substream model allows us to independently control delay, loss, and corruption so that information which requires rapid response and/or lower reliability is transmitted over the appropriate low-delay/low-loss substream, using the minimum wireless resources necessary. The multiple-substream approach also permits progressive resolution enhancement and asymptotic reliability.

Summary

The InfoPad system represents a new and unique approach to providing ubiquitous access to multimedia data. A unique aspect of the design is the movement of much of the computation that occurs on a portable computer onto machines in the backbone network. The major advantages of the InfoPad system design include:

- The user interface and application software execute on much faster and more capable machines.
- Less power is consumed in the portable terminal.
- Only error-resistant data such as bitmaps and audio samples are transmitted across the wireless links.

In traditional systems the speed of the portable system is dependent on the speed of the CPU and local electronics, whereas the usability of the InfoPad system is limited by the speed of its connection to the backbone network servers. As a result, the InfoPad model works best in environments where high-bandwidth connectivity is available.

This article has described the software infrastructure necessary to support the InfoPad system computation model. Special applications, type servers, and recognizers were developed for the InfoPad system. This software was designed to take advantage of the multimedia capabilities of the portable terminal and the additional computational resources available on the servers. The InfoNet system provides low-latency, high-bandwidth connectivity between the computation and the portable terminal. It also provides the routing and handoff support that allows pad users to roam freely. The performance measurements of the system show that this design is a viable alternative, especially in the indoor environment.

Acknowledgments

We thank Hari Balakrishnan for giving several comments and suggestions on earlier drafts of this article. We also thank the members of the InfoPad project team who helped design, implement, and test the system.

References

[1] A. Chandrakasan, A. Burstein, and R. Brodersen, "A Low Power Chipset for Portable Multimedia Applications," *Dig. Tech. Papers, Proc. ISSCC 1994*, San Francisco, CA, Feb. 1994, pp. 82–83.
[2] J. S. Eager, "Advances in Rechargeable Batteries Spark Product Innovation," *Proc. 1992 Silicon Valley Comp. Conf.*, Santa Clara, CA, Aug. 1992, pp. 243–53.
[3] S. Sheng, A Chandrakasan, and R. Brodersen, "Portable Multimedia Terminal," *IEEE Commun. Mag.*, vol. 30, no. 12, Dec. 1992, pp. 64–76.
[4] DECT92, "Radio Equipment and Systems. Digital European Cordless Telecommunications Common Interface, Part 2: Physical Layer," DI/RES 3001-2, European Telecommunications Standards Institute (ETSI), 1992.
[5] K. Keeton *et al.*, "Providing Connection-Oriented Service to Mobile Hosts," *Proc. 1993 USENIX Symp. on Mobile and Location-Independent Comp.*, Aug. 1993.
[6] S. E. Deering, "Multicast Routing in a Datagram Internetwork," Ph.D. thesis, Stanford Univ., Dec. 1991.
[7] S. Seshan, "Low latency handoffs in clellular data networks," Ph.D. Thesis, University of California at Berkeley, Dec. 1995.
[8] W. Scheifler and J. Gettys, *X Window System*, Bedford, MA: Digital Press, 1991.
[9] T. Levergood, "AudioFile: A Network-Transparent System for Distributed Audio Applications," *Proc. Summer 1993 USENIX Conf.*, June 1993.
[10] L.A. Rowe *et al.*, "MPEG Video in Software: Representation, Transmission and Playback," *Proc. IS&T/SPIE 1994 Int'l. Symp. on Elect. Imaging: Sci. and Tech.*, San Jose, CA, Feb. 1994.
[11] J. Kempf, "Integrating Handwriting Recognition into UNIX," *Proc. Summer 1993 USENIX Conf.*, 1993, pp. 187–204.
[12] N. Borenstein and N. Freed, "MIME (Multipurpose Internet Mail Extensions), Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies," Sept. 1993. RFC 1521.
[13] J. K. Ousterhout, *Tcl and the Tk Toolkit*, Reading, MA: Addison-Wesley, 1994.
[14] V. Kumar, *MBone: Interactive Multimedia on the Internet*, Indianapolis, IN: New Riders Publishing, 1996.
[15] L. W. Nagel and D. O. Pederson, "SPICE (Simulation Program with Inte-

grated Circuit Emphasis)," University of California, Electronics Research Laboratory, Memorandum ERL-M382, Berkely, CA, April 12, 1973.
[16] J. Wang, ftp://ftp.aud.alcatel.com/tcl, 1994.
[17] E. Amir, S. McCanne, and H. Zhang, "An Application Level Video Gateway," *Proc. ACM Multimedia '95*, San Francisco, CA, Nov. 1995.
[18] H. Balakrishnan *et al*., "Improving TCP/IP Performance over Wireless Networks," *Proc. 1st ACM Conf. on Mobile Computing and Networking*, Berkeley, CA, Nov. 1995.
[19] R. H. Katz, "Adaptation and Mobility in Wireless Information Systems," IEEE Personal Commun., vol. 1, no. 1, 1994.
[20] R. Han, L. C. Yun, and D. G. Messerschmitt, "Digital Video in a Fading Interference Wireless Environment," *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing*, Atlanta, GA, May 1996.
[21] L. C. Yun and D. G. Messerschmitt, "Variable Quality of Service in CDMA Systems by Statistical Power Control," *Proc. IEEE Int'l. Conf. on Commun.*, Seattle, WA, June 1995.
[22] A. Lao, A. J. Reason, and D. G. Messerschmitt, "Asynchronous Video Coding for Wireless Transport," *Proc. IEEE Workshop on Mobile Comp. Sys. and Apps.*, Santa Cruz, CA, June 1994.
[23] A. J. Reason *et al.*, "Asynchronous Video: Coordinated Video Coding and Transport for Heterogeneous Networks with Wireless Access," H. F. Korth and T. Imielinski, ed., *Mobile Computing*, Boston: Kluwer Academic, 1995].

Biography

SHANKAR NARAYANASWAMY was born in Singapore. He received his B.S., M.S., and Ph.D. degrees from the University of California at Berkeley in 1987, 1989, and 1996, respectively. His research interests include handwriting and speech recognition, user interfaces, and VLSI chip and system design. Future work directions include design of hardware and software platforms for multimedia and personal communications. His e-mail address is shankar@eecs.berkeley.edu, and his WWW home page is at http://infopad.eecs.berkeley.edu/shankar.

SRINIVASAN SESHAN received a B.S. in electrical engineering, an M.S. in computer science, and a Ph.D. in computer science from U. C. Berkeley in 1990, 1993, and 1995, respectively. Since 1995, he has been a research staff member at the IBM T. J. Watson Research Center. His research interests include computer networks, mobile computing, and distributed computing. His email address is srini@watson.ibm.com, and his WWW home page is at http://www.research.ibm.com/people/s/srini.

ELAN AMIR received a B.S. and an M.S. in computer science from U. C. Berkeley in 1993 and 1995, respectively. He has worked at the Lawrence Berkeley National Laboratory in the Networking Research Group and is currently working towards a Ph.D. in computer science at U. C. Berkeley in the field of real-time multimedia dissemination in heterogeneous environments. Other research interests include computer networks, mobile computing, and wireless communication. His e-mail address is elan@cs.berkeley.edu, and his homepage URL is http://www.cs. berkeley. edu/~elan.

ERIC BREWER is an assistant professor of computer science at U. C. Berkeley, and received his Ph.D. in computer science from Massachusetts Institute of Technology (MIT) in 1994. Interests include mobile and wireless computing (the InfoPad and Daedalus projects); scalable servers (the NOW and Inktomi projects); and application- and system-level security (the ISAAC project and Netscape security holes). Previous work includes multiprocessor-network software and topologies (Strata, metabutterflies) and high-performance multiprocessor simulation (Proteus).

ROBERT W. BRODERSEN received B.S. degrees in electrical engineering and mathematics at California State Polytechnic University; and from MIT he received the Engineers and M.S. degrees and a Ph.D. in electrical engineering. From 1972 to 1976, he was with the Central Research Laboratory at Texas Instruments. In 1976, he joined the Electrical Engineering and Computer Science faculty of U. C. Berkeley, where he is now a professor. In addition to teaching, he is involved in research involving new applications of integrated circuits, which is focused in the areas of low power design, wireless communications; and the applications made possible by these technologies.

FREDERICK BURGHARDT is a staff engineer with the InfoPad project in the Department of Electrical Engineering and Computer Science (EECS) at U. C. Berkeley. He received his B.S. and M.S. degrees in computer science from U.C. Berkeley in 1991 and 1994, respectively. His responsibilities include development of the InfoPad backbone network prototype (InfoNet) and support of InfoPad network research. Examples of this work include design and implementation of networked processes and specialized protocols for support of issues related to providing a computing environment for a wireless terminal, such as mobility and quality of service.

ANDREW BURSTEIN received a B.S. in electrical engineering from Cornell University in 1986. He received an M.S. in 1987 and a Ph.D. in 1996, both from U. C. Berkeley. He has previously worked at Bellcore in Morristown, New Jersey, and currently works at Hewlett Packard Laboratoriess in Palo Alto, California.

YUAN-CHI CHANG [S'94] was born in Taipei, Taiwan, Republic of China. He received the B.S. degree in electrical engineering from National Taiwan University in 1991. He is now a Ph.D student at the University of California, Berkeley. His research interests are in the areas of communication network QoS provisions and elastic source coding.

ARMANDO FOX received a B.S.E.E. from MIT. and an M.S.E.E. from the University of Illinois at Urbana-Champaign, and has worked as a CPU architect and validator for Intel Corp. He is currently a doctoral student in the InfoPad and Daedalus projects at the University of California at Berkeley, researching application-level support for adaptive proxy-based mobile computing, multimedia networks for mobile computing, and user interface issues related to mobile computing. He can be reached at fox@cs.berkeley.edu.

JEFFREY M. GILBERT was born in Waltham, Massachusetts, in 1971. He received his B.A. in computer science from Harvard University in 1993 and M.Phil. in computer speech and language processing from Cambridge University in 1994. He is currently pursuing a Ph.D. in electrical engineering at the University of California at Berkeley. He works on the InfoPad Project, researching compression techniques for transmitting color text/graphics and video over a wireless link for storage on low-power portable devices.

RICHARD HAN received his B.S. degree in electrical engineering with distinction from Stanford University in 1989. He received his M.S. in electrical engineering from UC Berkeley in 1991. He is currently completing his Ph.D. degree at UC Berkeley in the field of joint source-channel coding for wireless multimedia and graphics. His contribution to the InfoPad project includes development of robust graphics coding algorithms for InfoPad's split X windows server.

RANDY H. KATZ [F] is a leading researcher in computer systems design and implementation. His research experience has spanned numerous disciplines. He has written over 120 technical publications on CAD, database management, multiprocessor architectures, high-performance storage systems, and video server architectures. Professor Katz developed the concept of redundant arrays of inexpensive disks (RAID), now a $3 billion industry segment. Katz's recent research has focused on wireless communications and mobile computing applications. He is a Fellow of the ACM.

ALLAN C. LONG is a graduate student in computer science at UC Berkeley. For his M.S. project, he integrated a continuous media toolkit to the InfoPad to provide synchronized audio and video transport. He plans to begin on his Ph,D., researching interfaces that combine handwriting and speech recognition, this spring. He received his B.S. in computer science from the Unviersity of Virginia in 1992.

DAVID G. MESSERSCHMITT [F] is a professor and chair of the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. Current research interests include issues overlapping signal processing (especially video and graphics coding) and transport in broadband networks, network services and protocols for multimedia, and wireless multimedia computing. He received a B.S. degree from the University of Colorado, and an M.S. and Ph.D. from the University of Michigan, and is a Member of the National Academy of Engineering. He is currently serving on the Computer Sciences and Telecommunications Board of the National Research Council.

JAN M. RABAEY [F] received the E.E. and Ph.D degrees in applied sciences from the Katholieke Universiteit Leuven, Belgium in 1978 and 1983, respectively. From 1983 to 1985, he was a visiting research engineer with the University of California, Berkeley. From 1985 to 1987, he was a research manager at IMEC, Belgium. In 1987, he joined the faculty of the Electrical Engineering and Computer Science Department of the University of California, Berkeley, where he is now a professor. He has been a visiting professor at the University of Pavia (Italy) and Waseda University (Japan). Jan Rabaey has authored or co-authored more than 100 papers in the area of signal processing and design automation. He received numerous scientific awards, including the 1985 IEEE Transactions on Computer Aided Design Best Paper Award (Circuits and Systems Society), the 1989 Presidential Young Investigator award, and the 1994 Signal Processing Society Senior Award. He has served as associate editor of the *IEEE Journal of Solid State Circuits*. He is/has been on the program committee of the ISSCC, EDAC, ICCD, ICCAD, High Level Synthesis and VLSI Signal Processing conferences. He is the current chair of the VLSI Signal Processing Technical Committee of the Signal Processing Society. He is currently serving on the executive committee of the Design Automation Conference.