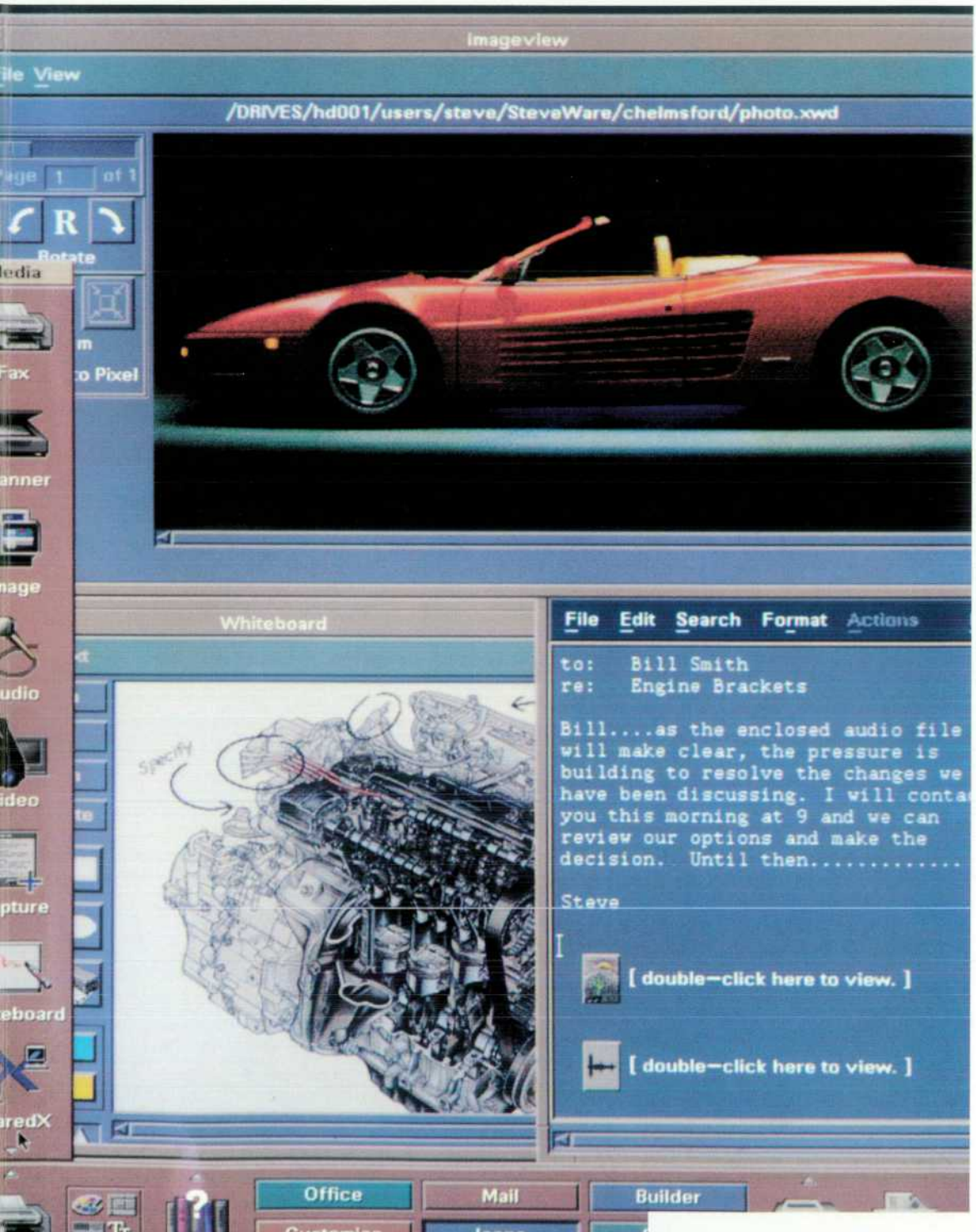
HEWLETT-PACKARD
JOURNAL
April 1994



HEWLETT-PACKARD
JOURNAL

April 1994 Volume 45 • Number 2

Articles

Departments

The Hewlett-Packard Journal is published bimonthly by the Hewlett-Packard Company to recognize technical contributions made by Hewlett-Packard (HP) personnel. While the information found in this publication is believed to be accurate, the Hewlett-Packard Company disclaims all warranties of merchantability and fitness for a particular purpose and all obligations and liabilities for damages, including but not limited to indirect, special, or consequential damages, attorney's and expert's fees, and court costs, arising out of or in connection with this publication.

Subscriptions: The Hewlett-Packard Journal is distributed free of charge to HP research, design and manufacturing engineering personnel, as well as to qualified non-HP individuals, libraries, and educational institutions. Please address subscription or change of address requests on printed letterhead (or include a business card) to the HP headquarters office in your country or to the HP address on the back cover. When submitting a change of address, please include your zip or postal code and a copy of your old label. Free subscriptions may not be available in all countries.

Submissions: Although articles in the Hewlett-Packard Journal are primarily authored by HP employees, articles from non-HP authors dealing with HP-related research or solutions to technical problems made possible by using HP equipment are also considered for publication. Please contact the Editor before submitting such articles. Also, the Hewlett-Packard Journal encourages technical discussions of the topics presented in recent articles and may publish letters expected to be of interest to readers. Letters should be brief, and are subject to editing by HP.

Please address inquiries, submissions, and requests to: Editor, Hewlett-Packard Journal, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A.

In this Issue

The 1990s may be remembered as the decade when multimedia capability became commonplace in computer technology, communications, and entertainment. The exact meaning of multimedia depends on who's using the word. On HP engineering workstations (HP 9000 Series 700 and 800 computers), one meaning of multimedia is HP MPower, a collection of multimedia hardware and software tools and applications that allow users to create, manipulate, and share textual information and nontextual information such as audio, image, graphics, and video data over a network.

As described in the article on page 10, on an HP MPower-equipped workstation the following services are available to the user: faxing, online documentation, scanning, image viewing, audio recording and playback, video in a window, window capture, whiteboard collaboration, real-time application sharing, and color graphics and PostScript™ printing. The development of HP MPower has been an evolutionary process, with new capabilities being developed as user needs changed and new technologies became available, and the product continues to evolve. The article on page 6 tells the story of this evolution and goes on to describe two very recently released HP MPower capabilities: digital video, or full-motion video with synchronized audio, and telephone functionality with a new HP MPower telephony component, HP TeleShare. These last two media types were added to HP MPower too late for articles on them to be prepared in time for this issue. We hope to include articles on their design in a future issue.

HP MPower, its various components, and its client/server architecture are introduced in the article on page 10. HP MPower's graphical user interface is the HP Visual User Environment, HP VUE. It's the subject of the article on page 20. The application sharing component of HP MPower is HP SharedX (page 23), a communication tool that extends the industry-standard X Window System so that two or more users at different workstations can share and interact with the same X-protocol-based applications almost as if they were at the same workstation. Existing X applications don't have to be changed to be shared with HP SharedX. Implementing this new application sharing technology in a heterogeneous computing environment, the designers discovered, poses many difficult design challenges, some of which don't have perfect solutions. A component of HP SharedX called Whiteboard (page 28) allows users to share a snapshot of a portion of a display and to annotate that snapshot.

Image files contain computer graphics and digital records of physical objects such as photographs, pages from books, and faxes. The HP Image Library (see page 37) contains image manipulation tools, compression and decompression functions, picture quality adjustment functions, and support for industry standards. Its functionality is used by several HP MPower components. For environments in which users have a multitude of printers to choose from, HP SharedPrint (page 44) provides a simple graphical interface that enables users to select a target printer and a set of options without many of the typical problems. The HP MPower fax utility (page 53) provides automatic dialing, transmission, and delivery of facsimile documents from a workstation.

HP MPower provides the hardware and software for recording and playing audio files over a network, incorporating audio in email, adding audio annotations to system files, and recording and playback using external devices such as tape recorders, CD players, and VCRs. HP MPower's audio functionality, application development tools, and hardware and software audio architecture are described in the article on page 62. Video technology in HP MPower is provided by a hardware/software component called HP VideoLive (page 68). It provides full-motion video in a movable, scalable window and works with existing HP graphics subsystems without degrading system or graphics performance.

HP MPower also provides a multimedia email, or electronic mail, facility (see page 71). The well-established processes of creating, sending, receiving, printing, and replying to email messages are maintained and applied to messages containing multimedia objects such as image and audio files.

Nearly 2000 online help topics are shipped with HP MPower. The HP online help system used by HP MPower and other HP applications is described in the article on page 79. On page 90, one of the designers of the HP help system advises application developers on the issues they may encounter in providing online help for their applications.

R.P. Dolan
Editor

Cover

A workstation screen showing the HP MPower media panel and the HP MPower applications ImageView, which provides capabilities for manipulating and viewing different types of images, MailEditor for creating multimedia email, and Whiteboard, which enables two or more users to collaborate on the same image from different workstations.

What's Ahead

Coming next are design articles on the HP 9000 Model T500 corporate business server and the SoftBench Message Connector. There will also be technical articles on the use of fuzzy logic to assign printed circuit assemblies to production machines and on cleanroom software.

Development of a Multimedia Product for HP Workstations

Providing multimedia capability on HP's workstations was an evolutionary process that was paced according to customer needs and the availability of quality multimedia hardware and software technology and low-cost workstations.

by Gary P. Rose, Jeffery T. Oesterle, Joseph E. Kasper, and Robert J. Hammond

Multimedia technology was a burgeoning market when HP's Workstation Group first looked at it in 1990. A lot of promise and exaggerated claims surrounded multimedia technology at the time. The question was how HP workstations could create a competitive advantage with the technology. The answer to this question resulted in HP MPower, a collection of multimedia tools and applications which are described in the articles in this issue beginning with an overview on page 10.

This paper will describe the development history of HP MPower and how it turned HP workstations from simply computational tools into media-rich information access and communication channels for business and industrial users.

The Start

Looking at the marketplace back in 1990 there were a number of application areas in which multimedia technology was being applied. Personal computers were being upgraded with CD-ROMs and sound cards, and typical multimedia application areas included presentations, computer-based training, and games. Workstations have difficulty competing with low-priced PCs for these markets. Since integrated networking capability was an advantage that workstations had over PCs at that time, we looked for markets that had distributed media requirements. We focused on two application segments: multimedia information management and real-time communication. The information-management market included document image management, work flow, and corporate training. The real-time communication market included workspace sharing, multimedia email, conference management, networked fax, telephone integration, and video teleconferencing.

We visited our customers to learn about challenges facing their businesses so that we could determine where we could offer solutions. A common theme we heard was that these companies needed to be more productive without significant increases in personnel. They were global companies that needed to align their teams on common objectives, and get them working together. Increasingly, they relied on distributed teams, alliances, and experts outside their company. The need for communication among these teams was critical to their success.

Communication between humans is more effective when it is natural. Media types such as recorded voice, pictures, and movies can add information to the communication that goes far beyond what traditional text can achieve. Facial expressions, body language, and tone of voice add cues to the meaning of the message. These cues help convey trust and understanding of what is being communicated. Multimedia computers can go far beyond traditional email in helping to facilitate communication, resulting in faster exchange and absorption of ideas. However, computer-assisted communication tools have to be easy to use and a natural part of the environment for them to be adopted by large numbers of people.

Although we wished that everyone owned an HP workstation, our customers did not have homogeneous HP environments. If the technologies we provided did not work with their existing equipment, then it would be difficult for them to deploy our products within their enterprises. Additionally, the importance of standards is very high in communication since they ensure that no one is excluded from a conversation because of the type of equipment they have.

We had three clear challenges for bringing multimedia to corporate offices. First, we had to deal with the limited networking capabilities of most existing environments. Second, the technology needed to be pervasive for people to use it. Finally, the technology needed to be very low-cost to be affordable for deployment within the enterprise.

End users were pulling the application developers into the multimedia arena. Thus, we needed to create desktop tools so users could immediately take advantage of multiple types of media without waiting for the applications to be developed. These tools also had to include examples of how to use the programming interfaces to the multimedia services so that application developers could immediately provide multimedia capability in their applications.

We wanted to leverage as much as possible the expertise within HP so we contacted numerous HP organizations. A PC-based collaborative multimedia project from HP Laboratories in Pinewood and Bristol, England was one of the pieces of research that helped guide us. Engineers in Bristol demonstrated that for distributed work groups trying to solve a range of tasks, a shared drawing space that allows multiple users to annotate a picture was very effective in improving productivity. Audio communication was considered the second most productive tool among these work groups. Surprisingly, seeing a video of the person they were

working with didn't improve productivity measurably. However, it is interesting that they perceived they were more productive when using video to show the object being discussed.

Our customer feedback was that although they all wanted to be able to do video conferencing from their desks, they did not have the network infrastructure in place. Also, when asked which media they would incorporate in their training and documentation, the answers were overwhelmingly in favor of images and audio. We felt that it was important to stage technologies for customer acceptance, and build up the capabilities over time. We decided to defer distributed digital video support until customers became comfortable with digital media over networks and implementations were cost-effective.

To keep the incremental costs for multimedia down we tried to implement as much as possible in software running on a PA-RISC CPU. This also allowed us to adapt our systems easily to new algorithms and standards, to provide access to our installed base, and to take advantage of new processor improvements.

1991—The Base Platform

Our customer feedback implied that the first technologies to be integrated should be image and audio. We asked customers about their imaging needs and found that while computers could display images, users typically had to run them through several conversion steps before their display program could put the image on the screen. Among graphics products there was a wide range of image formats and frame buffer pixel depths. This made image display inconsistent from machine to machine. Another problem was that the screen would turn funny colors when more than one image was displayed at a time because of the lack of color map sharing.

We addressed these problems with an image library and tried to make images as easy to use as text and graphics. We integrated image and audio libraries into the HP-UX* operating system so that applications would have an installed base for their functionality. There were no standards available for programming interfaces, so we modeled the interfaces to feel like X windows, which is a paradigm familiar to our application developers. Rather than creating HP file formats, common formats from the PC and Apple Macintosh worlds were used and conversion services were provided to import and export data from these platforms. We used algorithm expertise from HP Laboratories and the CPU power of our systems to include compression and decompression of images using the JPEG (Joint Photographic Expert Group) standard, which allows images to be useful on low-end machines with small disks. The image library was designed as an extensible pipeline architecture that would allow applications to add new file types or special operations.

Our approach to audio support was to integrate audio on the motherboard of our workstations. Instead of taking the traditional approach of providing a DSP (digital signal processor) for moving the audio to the CODEC (coder/decoder), we use the main processor. This not only saves the cost of the DSP but also the dedicated memory for the DSP and other support logic. The PA-RISC processor is much faster than commercial DSPs, and it allows more complex functions to be applied to media streams.

We felt it was important to develop small applications such as an audio editor that would provide end user tools so there would be market demand for the technologies. We also gave away the source code for these small applications so that developers would have working examples to start with when they developed their own applications.

The audio and image library were packaged with our X window sharing product HP SharedX, making up our first multimedia offering. The audio library, the image library, and HP SharedX are described on pages 62, 37, and 23 respectively.

1992—Media I/O

In 1992 we decided to make our existing technologies more useful and postponed digital video. We felt we could bring our customers more value by leveraging the strengths HP had in computer products and integrating those products with the base tools. We ported the HP ScanJet IIc from Microsoft® Windows to the X Window System to provide a way to get images into the workstation. We created a product called HP SharedPrint to allow a multitude of image formats to be printed on the wide range of PCL and PostScript™ printers available. We added fax technology so that users could have another way to communicate with images. This would also allow communication with people outside their normal networking environment. We collaborated with third-party vendors to provide hardware for video in a window and to allow users to capture digitized frames from the video. HP SharedPrint, HP MPower fax, and our first video offering are described on pages 44, 53, and 68 respectively.

We upgraded the audio that is built onto the CPU board to CD quality to anticipate low-cost speech recognition, text-to-speech capability, and computer-based training. The HP-UX elm mailer was integrated with the new media data types to handle compound document mail messages using the internet standard MIME (Multipurpose Internet Mail Extensions). To improve the usability of the system we did extensive up-front task analysis. We determined how the tools would be used to accomplish different tasks and worked to eliminate the number of steps users needed to succeed at those tasks. We made the user interfaces appear more consistent among the different tools. We used HP SharedX to replicate our graphical user interfaces and solicited feedback from the different HP organizations developing components for HP MPower and HP VUE (Visual User Environment) 2.0 customers. The HP VUE team worked closely with us to integrate the media and collaborative tools into the control panel of the HP VUE 3.0 control panel. We delivered this collaborative user environment to the market under the name HP MPower 1.0.

HP VUE 3.0 and the new elm editor are described on pages 20 and 71 respectively.

1993—Ready for Video

In 1993 we improved HP MPower in three dimensions by adding digital video, integrating telephony, and dramatically improving the flexibility for configuring the client/server

environment so that fax and print servers can reside on different machines. These features became HP MPower 2.0.

Digital Video in HP MPower

Recent advances in computer-processing speed and video-compression techniques have made it possible to combine full-motion video and synchronized audio into a form of computer data. This data, known as digital video, can be delivered over standard computer and telecommunication networks and can be integrated into multimedia applications such as computer-based training programs.

In the computer-based training market, there is typically a small number of authors and a large number of people who use this form of training. Our goal was to deliver cost-effective digital video playback for desktop computer-based training. We worked with HP Laboratories and the HP 9000 Model 712 team to integrate the video playback algorithms tightly into the PA-RISC 7100LC chip. The graphics team provided new blithering (dithering and visual blending) algorithms and media-oriented frame buffer access modes that greatly assist in the rendering performance, giving the appearance of a 24-bit system with the cost of an 8-bit system.

Standards-Based File Format. HP's digital video implementation supports the MPEG-1 (Moving Pictures Expert Group) file format. MPEG-1 is an internationally recognized standard for compressing synchronized audio and video data. MPEG-1 maintains a high-quality image (comparable to VHS tape) while supporting compression ratios up to 200:1.

Key Benefits. HP MPower users can play MPEG-1 movies on any HP 9000 Series 700 workstation without additional hardware. The new HP 9000 Model 712 workstation provides exceptional price/performance value for playing video because of instruction set enhancements to the Model 712's PA-RISC chip and enhancements to the graphics subsystem.

Since MPEG-1 movies are a form of digital data they can be transferred to other users via email or standard HP-UX commands such as ftp or uucp.

Digital Video Components. HP MPower 2.0 has two digital video software components: the video player and the video converter. The video player software plays MPEG-1 movie files with or without audio (see Fig. 1). The user can adjust the size of the window and adjust the audio and video qualities. Any frame in the video can be examined, and play forward or reverse capabilities are also available. Video frames can be captured and saved as TIFF, JFIF, Xbm, or Xwd images. Images can also be printed directly from the application.

The video conversion utility converts JPEG movie files to MPEG-1 format. JPEG (Joint Photographic Expert Group) is an internationally recognized standard that deals with the compression of still images.

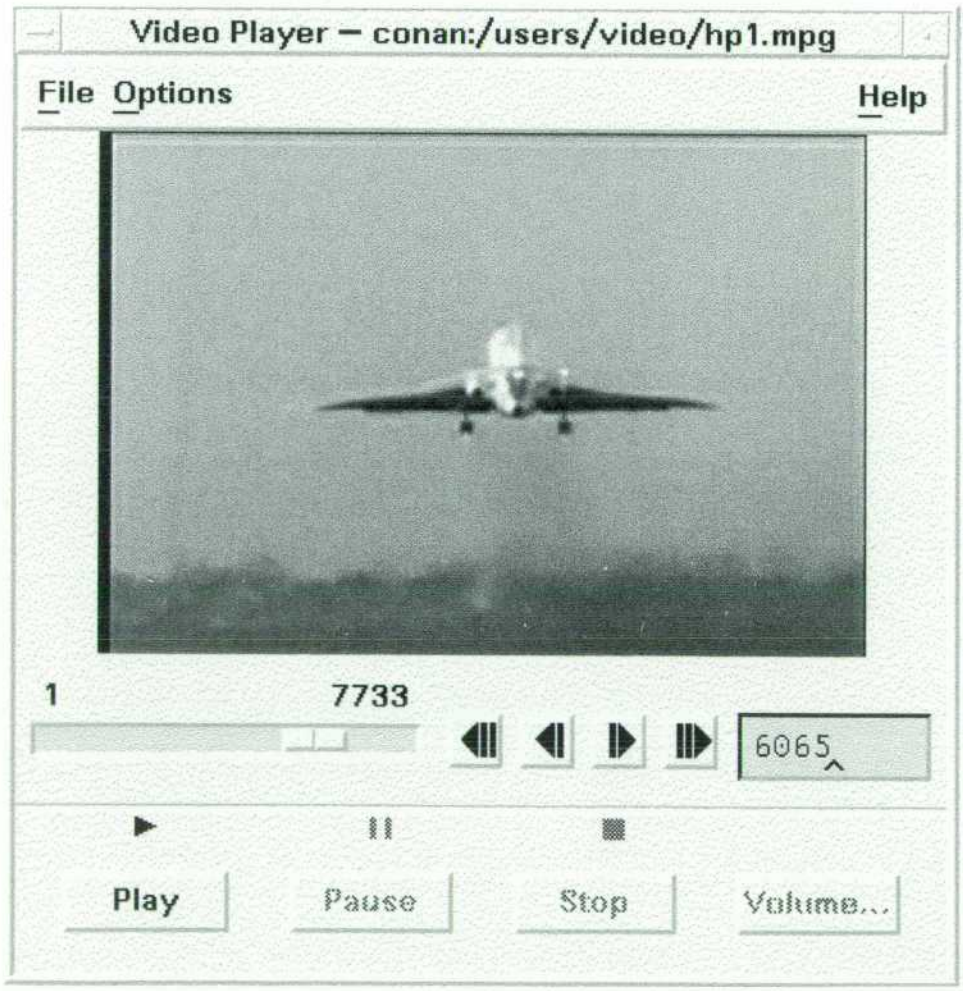


Fig. 1. The digital video player playing an MPEG-1 movie file.

MPEG-1 movies can be obtained by capturing video data from an external source such as a video camera and storing it in JPEG movie files.† The JPEG files can then be converted to MPEG-1 format with the video conversion utility. Alternatively, customers can use a service bureau to convert their video material to MPEG-1 format.

Telephony

The integration of telephone functionality on a workstation provides users with a powerful communication tool that enhances the use of both the telephone and the computer. The telephone becomes easier to use because the computer can take care of the details of telephone use such as special function buttons, volume control, finding and dialing phone numbers, and tracking telephone use. The computer becomes a more effective communication and collaboration tool. Also, with telephone access, users can send faxes from their desktop, share their computer audio over the telephone, and get caller information from a database based on the caller identifier (e.g., telephone number).

HP's telephony product, or HP TeleShare, provides a two-line telephony card for HP's 9000 Model 712 workstation. The HP TeleShare card is an optional daughtercard, with two complete analog†† telephone line interfaces. Each telephone line has a digital signal processor to provide data and fax modem support and to handle audio mixing for voice mode use. Having two telephone lines provides the capability to place a telephone call using one line while setting up a data or fax connection on the other line.

HP TeleShare also includes a telephone application that provides users with access to various telephone functions from their workstations. For switching the mode of each telephone line between data modem, fax modem, and voice there is a small control application that controls the mode of each HP TeleShare line and reflects any changes in mode to the user. Fax functionality is provided through a single-user configuration of the HP MPower fax facility, while data modem functionality can be accessed through the user's favorite data communications package such as kermit or cu, provided they are configured to use the HP TeleShare card as a modem.

HP TeleShare has direct access to the HP MPower audio subsystem on the workstation, which is what makes it possible for a workstation with the audio headset to be used as a full-function telephone. This also makes it possible to share computer-generated audio over the telephone line and to record telephone conversations into computer audio files for later reference. Because telephone audio is low-quality, HP TeleShare provides services to deal with quality levels. The audio server automatically resamples the computer audio so that the user is not constrained by this restriction. This makes it possible to play CD-quality samples over the telephone line or to record from the telephone line into a CD-quality sample file.

Applications. Two OSF/Motif applications are provided with HP TeleShare. The first is called teleshare, which provides a graphical user interface to the telephone functions provided by the product. These functions include a telephone keypad, volume and hook controls, forwarding buttons, programmable speed dial keys, and a display area for incoming caller-identifier information.

The second application, called telctrl, provides control and status information on the media modes (fax, data, voice) for the HP TeleShare telephone lines (described below). There is also a helpful graphical setup and configuration program to help the system administrator configure the HP TeleShare product properly.

Fax and Data Modem Lines. HP TeleShare can function as a fax or data modem in addition to serving as a full-function telephone. The telctrl application allows control of the current mode of each of the HP TeleShare telephone lines, as well as reflecting any changes in the mode. The mode can also be changed automatically when a modem application opens a connection to the port supplied for interfacing to HP TeleShare's modem functionality. Only one line at a time can be used as a modem, but the other telephone line would be available for voice mode use.

A single-user configuration of the HP MPower fax product is shipped with HP TeleShare to provide support for fax functionality.

Conclusion

Enhancements or additions to the HP MPower product will be guided by our ability to leverage HP and external multimedia tools and technologies and integrate them into the product to reduce cost and to take advantage of HP's distributed computing and object-oriented design expertise. We will continue to listen to our customers' needs and provide frameworks that will allow tighter integration of the parts to improve usability. Our internal use of the collaborative tools for our own communications with remote experts and teams both inside and outside of HP will provide us with additional insight into communication needs for the future.

† A third-party video card must be used to capture JPEG movies.

†† Analog telephone line refers to the traditional, Plain-Old-Telephone-System (POTS) telephone lines, as opposed to ISDN (integrated-services digital network) or ISDN-like proprietary digital telephone systems.

HP MPower: A Collaborative Multimedia Environment

Multimedia capability on a workstation enables users to interact with their applications and communicate with others in a variety of formats (textual and nontextual). HP MPower provides an environment in which users have easy access to the multimedia facilities at their workstations, and application developers can easily add new multimedia tools.

by William R. Yoder

Imagine being able to have a project team meeting in which the participants are widely dispersed but are able to collaborate from their desktop workstations as if they were all in the same room. To carry out such an electronic meeting, the participating workstations must provide the facilities that allow users to create, manipulate, and share textual and nontextual information such as audio, image, and video data over a network.

HP MPower provides workstation conferencing and the collaborative sharing capabilities mentioned above. Unlike video teleconferencing, which requires a significant hardware and networking investment, HP MPower is a low-cost software product that works with today's workstations and networks. HP MPower offers a full range of multimedia types such as audio, image, graphics, video, and text (Fig. 1), with five ways of sharing information: print, mail, fax, whiteboard, and real-time application sharing. HP MPower offers access to this set of multimedia tools through the HP VUE 3.0 graphical user interface.

HP MPower is currently supported on the HP 9000 Series 700 and 800 workstations and HP X stations.

The Media-Equipped Knowledge Worker

A typical knowledge worker uses a workstation to process information in the form of documents, spreadsheets, graphics presentations, and so on. In addition to these items, the media-equipped knowledge worker has access to sound clips, video frames, scanned images, faxes, and other media objects. Whether among a local team or among colleagues who are scattered geographically, the media-equipped knowledge worker benefits by sharing high-fidelity information at a high bandwidth.

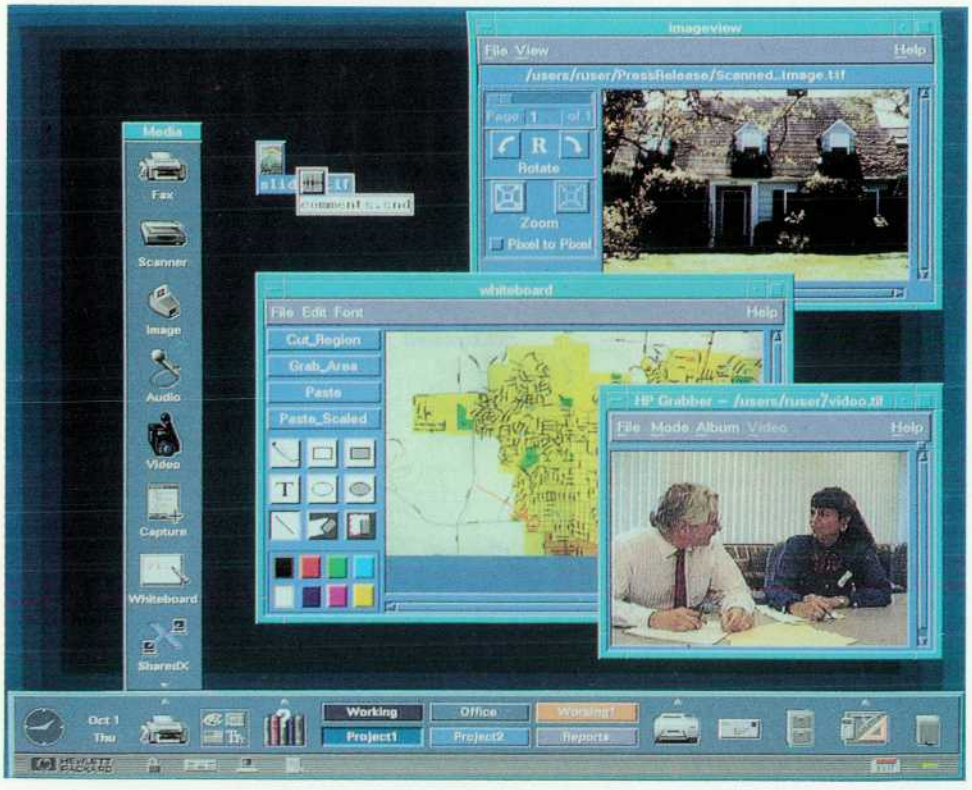


Fig. 1. A typical HP MPower display showing windows open for whiteboard, images, and video.

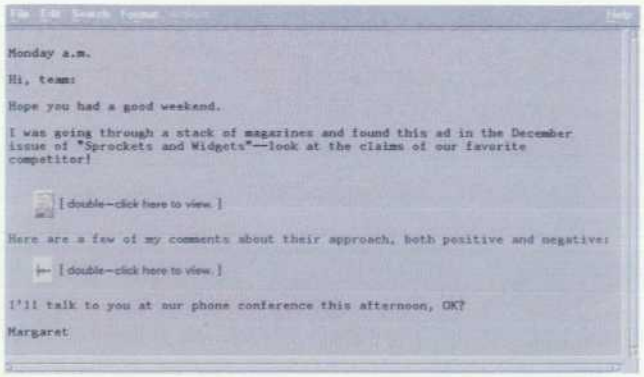


Fig. 2. Multimedia mail composing.

Workstations and PCs capable of providing multimedia access have until recently existed as islands of technology, only good for standalone applications. A developer logs into such a workstation and creates, say, a training module that end users can access only at the isolated workstation. By combining the power of media technology with networked systems running the HP-UX* operating system, HP MPower enables users to collaborate effectively via a workstation medium.

For example, suppose Margaret wants to send her colleagues a document consisting of text, a scanned image of a competitor's magazine ad, and a voice clip commenting on the article. She presses the mail button on the HP VUE front panel shown in Fig. 1 to compose the mail message shown in Fig. 2, presses the scanner button on her HP MPower media panel to scan in the ad, and then presses the audio button to record her comments. Finally, she drags and drops the scanned image and voice clip into her mail message and sends it on its way.

As another example of using this media-equipped workstation, consider Jeffrey who wants to work on a CAD drawing with his colleagues in Colorado and Washington. From the HP VUE file manager, he drops the drawing into a whiteboard window on his workstation, calls his colleagues on the telephone, and uses the whiteboard window to interact and collaborate with his colleagues. The whiteboard and the software for sharing windows are discussed in the article on page 23.

The HP MPower System

On a fully-equipped HP MPower-enabled workstation the following services are available to the user:

- Faxing
- Online Documentation
- Scanning
- Image Viewing
- Audio Recording and Playback
- Video-in-a-Window
- Window Capture
- Whiteboard Collaboration
- Application Sharing
- Color Graphics and PostScript™ Printing.

Many of the services listed above are accessible from the HP MPower media panel shown in Fig. 3. Some of the other HP MPower features accessible from the front panel include:

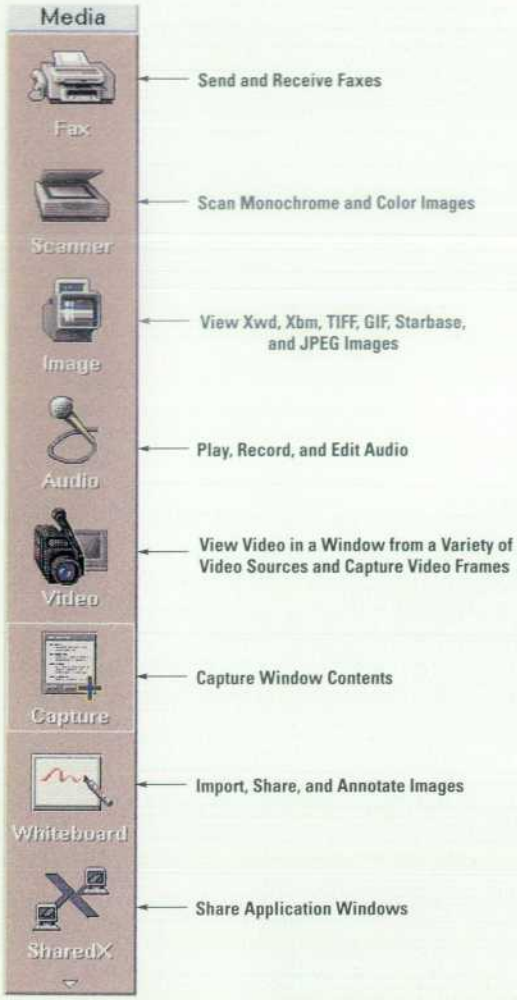


Fig. 3. HP MPower media panel.

- Audio control for adjusting global audio output devices
- Help control for accessing system-wide online documentation
- Print control for printing text and graphics, managing print requests, and administering printers
- Mail control for reading and composing plain text and media-embedded mail messages.

Hardware Components. The basic HP MPower workstation consists of a high-resolution display, a keyboard, and a mouse. For a fully-equipped HP MPower multimedia workstation the other hardware components include:

- Built-in 8-bit or 16-bit audio with speaker or plug-in headset
- External scanner with SCSI interface
- External fax modem (or modems) with serial interface
- EISA-based video card
- A variety of serial and parallel printers.

Software Components. HP MPower software consists of a number of tightly integrated media tools coupled to the HP VUE 3.0 user environment with interprocess communication mechanisms for distributed processing. HP VUE 3.0 and HP MPower are peers in the software hierarchy (see Fig. 4). When the user selects an HP MPower media object (e.g., audio file) from the HP VUE 3.0 display, HP VUE hands control over to HP MPower to take the appropriate action on the object.

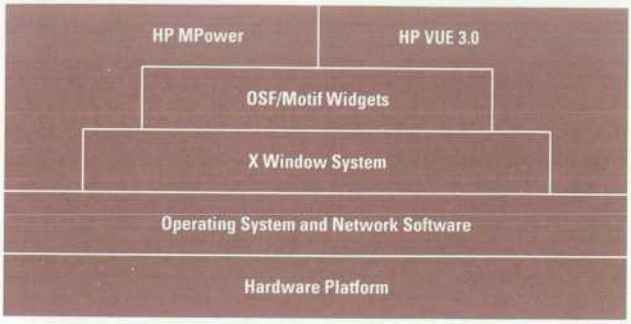


Fig. 4. The software hierarchical relationship between HP VUE 3.0 and HP MPower. HP VUE 3.0 provides user interface and desktop services for the look and style of HP MPower media objects, and HP MPower provides the actions associated with a particular media object.

A typical media tool consists of an OSF/Motif-based client application, its run-time libraries, a backend server process, and the appropriate device drivers (see Fig. 5). At the lowest level, the device drivers control the hardware. For example, the VideoLive card uses an X-server extension to access the frame buffer. This X-server extension enables direct hardware access to the frame buffer, so that the VideoLive client can manipulate 24-bit 640-by-480-pixel images within the context of an 8-bit root window. Media server components are described in more detail later in this article, and the VideoLive card is described in the article on page 68.

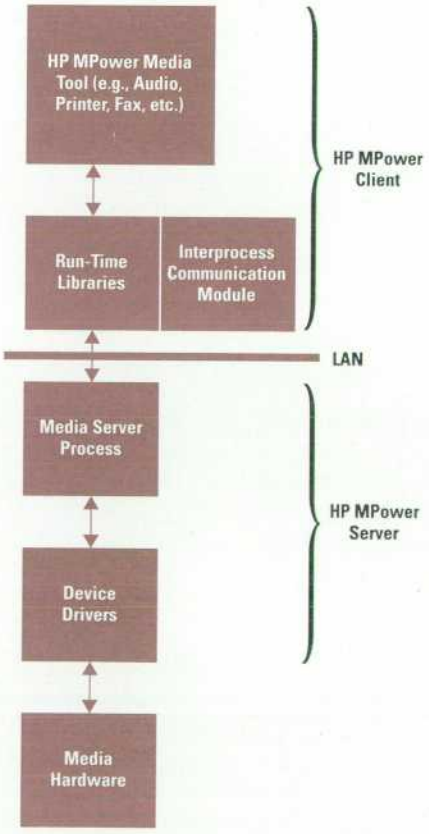


Fig. 5. The architecture for a distributed multimedia application.

Important HP MPower run-time libraries include the image and audio libraries, which are described in the articles on pages 37 and 62, respectively.

The User Interface

From a user's point of view, the HP MPower workstation consists of an integrated set of tools and their associated media objects. HP MPower provides facilities that enable the user to:

- Create media objects like a video frame sequence
- Browse objects such as an incoming fax
- Edit objects such as an audio track
- Share objects such as a workstation window.

The appearance and behavior of HP MPower are derived from the OSF/Motif style guide and from the HP VUE desktop. For example:

- Users can double-click to invoke actions, as in playing an audio file
- Users can drag and drop media objects on HP MPower tools, as in dragging and dropping an image file on the fax composer.

HP MPower tools are mouse-driven with pushbuttons, pull-down menus, and dialog boxes.

The HP VUE 3.0 interface is described on page 20.

Objects and Actions

The file-typing mechanism used by HP VUE is extended to media objects. For example, PostScript files are denoted by a .ps suffix appended to the base file name (e.g., Article.ps). The HP MPower media tools such as the audio editor ensure that files are created with the appropriate suffix.

Each media object has certain allowable actions or methods. For example, for audio files appropriate actions include Play, Edit, and Mail, and for image files appropriate actions include View, Print, Mail, and Fax. Table I lists the objects and actions supported in HP MPower 1.0. These HP MPower actions extend the predefined HP VUE 3.0 objects and actions.

Online Documentation

Extensive online documentation is provided with the HP MPower system. Built on the HP VUE 3.0 help system, HP MPower online documentation includes component level documentation (e.g., help on the fax composer) and system level documentation (e.g., the "Welcome to HP MPower" chapter).

Top level indexes provide users with easy access to all the help volumes on their system. The many hyperlinks† among topics allow users to browse hundreds of pages of task-oriented and reference material, which may or may not be related to the task they are performing. Another type of help called item help enables users to find answers as they use the media tools in the context of the task they are performing.

With the exception of a minimal set of introductory online documentation, all help text is installed on the HP MPower server to conserve client disk space. More about HP MPower help is covered in the articles on pages 79 and 90.

† Hyperlinks are navigation pointers to related pieces of information. The online help article on page 90 provides more information about hyperlinks.

Table I
HP MPower Objects and Actions

Object	File Suffix	Appropriate Actions in HP MPower
SUNFILE	.au	Create, Edit, Mail, Play
NEXTFILE	.snd	Create, Edit, Mail, Play
WAVFILE	.wav	Create, Edit, Mail, Play
L16FILE	.l16	Create, Edit, Mail, Play
L8FILE	.l8	Create, Edit, Mail, Play
Lo8FILE	.lo8	Create, Edit, Mail, Play
ALFILE	.al	Create, Edit, Mail, Play
UFILE	.u	Create, Edit, Mail, Play
PS	.ps	View, Print, Mail, Fax
EPS	.eps	View, Print, Mail, Fax
TIFF	.tif, .tiff	View, Print, Create, Mail, Fax
GIF	.gif	View, Print, Mail, Fax
JPG	.jpg, .jpeg	View, Print, Create, Mail, Fax
BMF	.bmf	View, Print, Mail, Fax
BM	.xbm, .bm	View, Print, Create, Edit, Mail, Fax
PM	.xpm, .pm	View, Print, Create, Edit, Mail, Fax
XWD	.xwd, .wd	View, Print, Create, Mail, Fax
MIME	.mim	View, Print, Create, Edit, Mail, Fax
Unknown	.unk	Create, Mail
Data/Text	(any)	View, Print, Create, Edit, Mail, Fax

Client/Server Architecture

HP MPower is shipped in a client/server configuration, allowing applications and data to be distributed across a networked computing environment. In a client/server architecture, programs and data are split across the network according to each machine's capabilities. The term server refers to a program offering a service such as faxing or printing. The term client refers to a program requesting a service, such as the fax composer or the HP SharedPrint client.

In today's client/server world, many services are typically concentrated on a powerful central system, which is termed a server system. For example, the HP MPower server offers built-in fax, mail, font, help, print, and HP VUE services. The functionality available locally on the user's desktop is collected on what we call the HP MPower client.

Advantages of the HP MPower client/server architecture include:

- Distributed processing
- Maximum performance measured by interactive response times and load balancing
- Minimum cost-per-seat realized by RAM and disk savings
- Scalability in that when new clients are added, the system administrator can either add new servers or simply add RAM and disk to existing servers.

Fig. 6 shows a typical HP MPower client/server configuration. In this configuration a client can fax a document via an HP MPower server to another client (see ① in Fig. 6). Likewise a client can send a document to an HP MPower server (provided it offers print services) to be printed on a specific printer (② in Fig. 6). Typically, an HP MPower server provides fax, printer, font, online help, and user interface services. The HP MPower client provides local image processing, audio, video, and display services. Applications, such as spreadsheets, can run on the server, on the client, or on dedicated application servers.

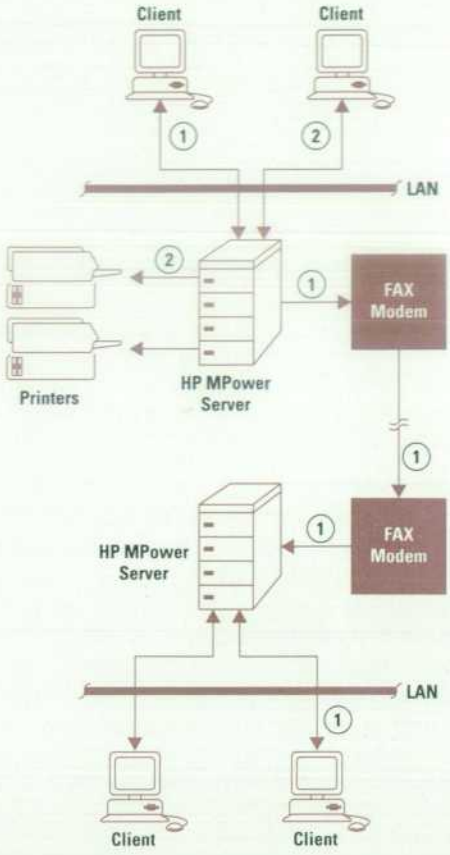


Fig. 6. Faxing a document between two clients connected to different HP MPower servers.

HP MPower services can be split among a variety of machines in extremely flexible configurations. For example, HP SharedPrint servers are installed on whatever machines have printers physically connected, the fax server can run on a machine different from the HP MPower server, and there can be two or more HP VUE servers providing login, file, and window management services for a large group of users. However, in arranging services in such a manner an extra burden is put on the system installer and network administrator.

For simplicity, the HP MPower server by default runs all the HP MPower services. The HP MPower client on the user's desktop runs whatever productivity, multimedia, or user interface applications it can offload from the HP MPower server.

Server Processes

HP MPower server processes include the device drivers and the interprocess communication software shown in Fig. 5. The server processes in the HP MPower 1.0 network include:

- An X11R5 display server with HP SharedX and video extensions

- An audio server that manages local audio hardware
- A font server that manages the fonts on the HP MPower server
- A fax server that manages local fax modems
- A print server that manages local printers.

These server processes enable client applications to access a serially reusable resource attached to a given host. The font server enables the HP MPower server to service font requests from all applications, affording significant disk savings. The fax server handles file conversions (e.g., converting from PostScript to fax-file format), call routing, administrative databases, and incoming and outgoing telephone connections. The print server employs a variety of filters to convert popular imaging formats to a given printer's native language.

Descriptions of the audio, fax, and print server processes are covered in the articles on pages 62, 53, and 44, respectively. HP SharedX, which is a tool for sharing windows, is described in the article on page 23.

Interprocess Communication Mechanisms

HP MPower employs a variety of interprocess communication mechanisms to enable its asynchronous, distributed processes to communicate and cooperate. UNIX* domain and internet sockets provide most of the substrate, enabling remote procedure calls and event-driven protocols. Helper processes include a remote invocation daemon for launching distributed applications, a broadcast message server for passing simple strings, a location broker daemon for establishing connections, and other standard UNIX services (X server, name server, remote print daemon, NFS-mount daemon, and a mail transport mechanism).

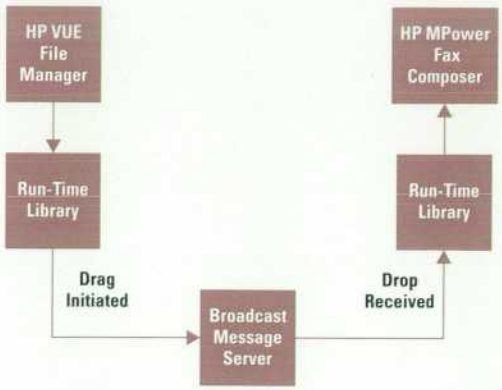


Fig. 7. The broadcast message server enables the fax composer to accept a dropped file from the HP VUE 3.0 file manager.

For example, in extending the HP VUE 3.0 drag and drop mechanism to the HP MPower tools, the broadcast message server (BMS) provides the communication link (see Fig. 7).

Desktop Configurations

HP MPower provides four preconfigured desktop clients as shown in Fig. 8. For each configuration, Fig. 8 indicates the

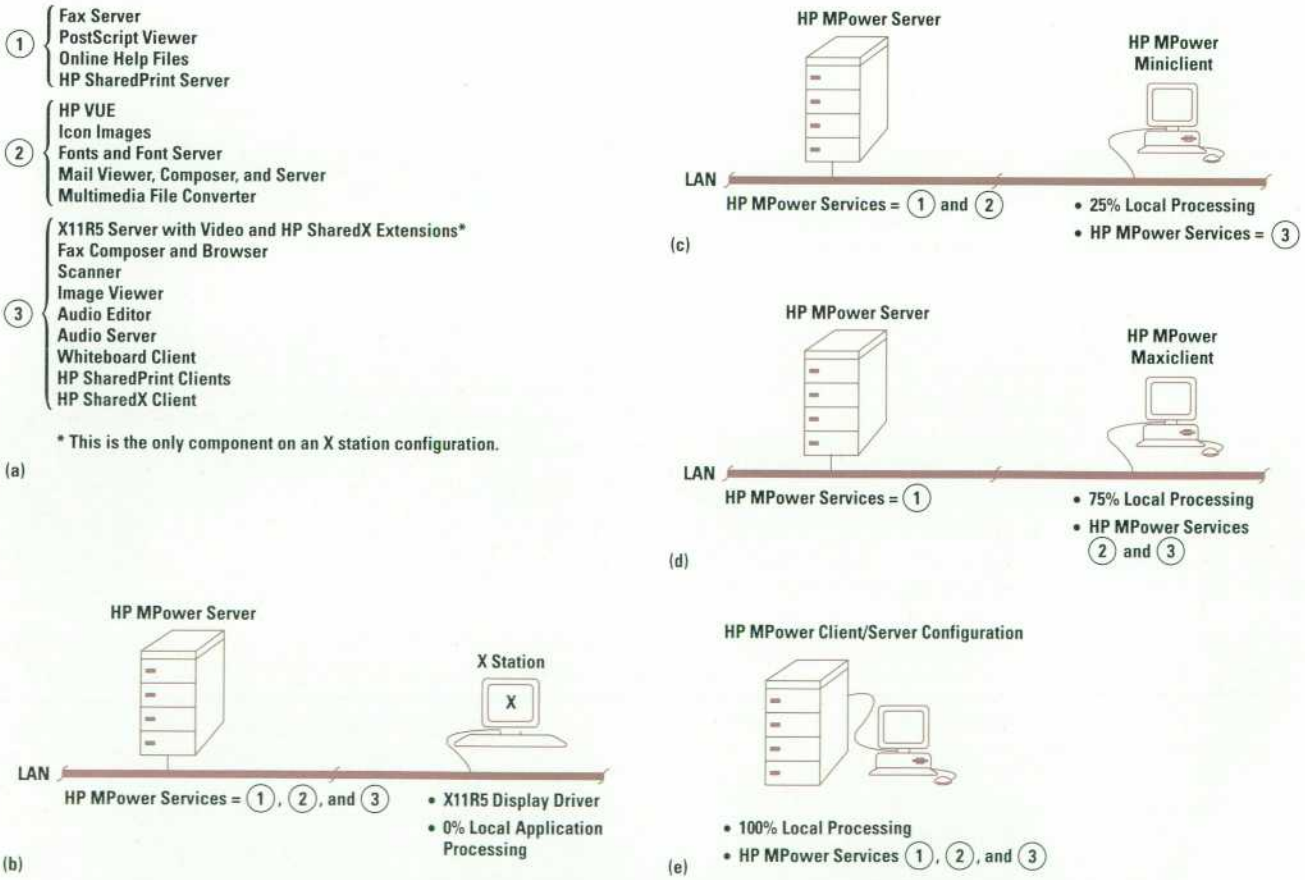


Fig. 8. Different desktop configurations provided with HP MPower. (a) HP MPower services provided by clients and servers. (b) X station configuration. (c) Miniclient configuration. (d) Maxiclient configuration, (e) Client-on-server configuration.

approximate share of the HP MPower processing load that occurs locally on the desktop client before any other applications are started. Fig. 8a shows the HP MPower services distributed among the configurations shown in Figs. 8b to 8e.

X Station Configuration. In the X station configuration the user runs HP VUE 3.0 and the media services totally from the HP MPower server (Fig. 8b). There is no local processing, other than the display server portion of the X11R5 window display system.† HP 9000 Series 300 and 400 workstations and X stations and workstations from other vendors can function as HP MPower X stations. These stations will run the HP MPower software entirely on the HP MPower server (including the client services), but can run other applications locally on the workstation.

For more about X stations (or X terminals) see "X Stations in HP MPower" on page 16.

Note that the media tool software architecture shown in Fig. 5 is not applicable to the X station desktop configuration because with the exception of the display and audio driver software all software runs on the HP MPower server.

Miniclient Configuration. On an HP MPower miniclient most media services, such as audio and imaging, run locally (Fig. 8c). The user's HP VUE session, including the file manager and the window manager, runs on the HP MPower server.

The miniclient configuration takes advantage of local HP-PA RISC processing power for imaging operations, such as rotation, scaling, and contrast.

Maxiclient Configuration. On an HP MPower maxiclient the HP VUE user interface and most media services can be run locally (Fig. 8d). The advantage here is that the dependence on the HP MPower server for the desktop user interface is removed.

Client-on-Server Configuration. This configuration uses a fully loaded workstation, running both server and client HP MPower software (Fig. 8e). Essentially, it offers standalone media services suitable for both networked and nonnetworked environments.

This configuration is easiest to configure and administer, but it is the least cost-effective solution. Also, it is limited to computers that support bitmap displays.

Network Home

To provide a consistent environment for users whose data files reside on one machine and applications on another, HP MPower implements a network home environment. This environment provides the user with a view of files that is consistent across all machines in the HP MPower network. The user sees the same colors, fonts, home directory, and so on regardless of the HP MPower client or X terminal on which a session is started.

To set up the home environment, system administrators have three options for locating users' personal data (i.e., $HOME directories):

- Leave the $HOME directory on the user's desktop workstation
- Place the $HOME directory on the HP MPower server
- Place the $HOME directory on an alternate file server.

Installation and Configuration

Because of its complex interprocess interactions and client/server architecture, HP MPower depends heavily on the functionality of the HP-UX operating system. For example, HP-UX scripts are used to customize HP MPower during initial installation when the HP Instant Ignition†† process is running. Scripts are also used to add, delete, and reconfigure HP MPower clients.

Two of the most important scripts, which are run at instant ignition boot time, are responsible for setting up the HP MPower server and HP MPower clients. The server configuration script (setup_server) performs the following functions of the server system:

- Starts the network file system (NFS) for remote file access
- Starts the NFS automounter service to provide transparent access to remote file systems
- Starts the network computing system (NCS) to support remote printing, faxing, and audio
- Starts the X11 font server so that HP MPower clients can obtain their fonts from the HP MPower server system
- Starts the sendmail daemon so that users can send and receive multimedia mail.

The setup_client configuration script, which runs on the HP MPower client, performs the following functions:

- Enables NFS and automounter
- Sets up links between the client and server for the local client file system
- Establishes the fax, HP SharedPrint, and HP VUE connections to the HP MPower server
- Enables the drag and drop capability between server and client.

The system administrator can add or remove other clients at any time by running a simple admin_server script on the HP MPower server.

Session Startup and Login

At later system boots (after the initial installation described above), the /etc/src.sh script sets certain key global environment variables, such as the HP VUE server. On the HP MPower server, the /etc/rc file starts the fax server and the font server processes. Other boot-time scripts start the NFS, automount, and HP VUE login processes.

When a the user logs in through the graphical HP MPower welcome screen, other variables such as the user's audio host, help path, and network home are established.

Testing

Because of the number of software components and hardware configurations, testing HP MPower was a daunting task. We concentrated on the configurations that would be most popular, as directed by our product marketing team, with particular emphasis on the HP 9000 Models 712 and 715 machines configured as standalone desktop clients. In the course of the project, we used more than 20 integration and test machines internally to verify software installation and configuration.

† The new HP ENVIZEX X stations offer a local flexible disk drive, audio, printing, and scanning.

†† See "The HP Instant Ignition Program" on page 17.

X Stations in HP MPower

The X station (or X terminal) is a product optimized to run X Window System server software. X stations were developed when the X Window System was established as a standard distributed windowing system for the UNIX operating system. Three major factors accelerated the acceptance of X stations in the market:

- Emergence of the client/server model of computing
- Dramatic increases in processor compute power
- Improvements in networking technology.

The X station is a network-based display device that uses X protocol over a local area network (LAN) to communicate with the host. The programs specially written for X (called clients) run on the host but display their output on the X station. X stations are also able to run programs locally. The programs that run locally on the X stations are referred to as local clients. There are three major classes of local clients: local window managers, local terminal emulators, and local utilities.

X stations cannot operate without a host because they use the compute power, memory, and disk space of the host machine.

X Stations versus Workstations

The X station is a complementary product to the workstation in that it offers the look, feel, sound, and graphics performance of a workstation, but at a much lower price. Typically, the X station costs about half what the comparable workstation costs, while providing workstation-like graphics performance. X stations allow multiple users to access the power of a modern workstation (host), help to make better use of the compute power and disk space available on the network, and simplify system administration. X stations are not suitable for two types of users:

- Power users that run simulation and modeling programs
- 3D graphics users.

The reasons X stations are less expensive than workstations include:

- They use a low-cost embedded graphic controller as CPU. A general-purpose processor used in a workstation is much more expensive.
- They need much less memory to perform the same tasks since they have a compact, real-time, UNIX-system-like operating system that uses only a small fraction of the DRAM required for a complete UNIX operating system.
- Most of their electronic circuitry is integrated into application-specific integrated circuits (ASICs) to reduce cost even further.
- They use less power, making their power supplies less expensive.
- They do not have any hard disks.

Configuring X Stations in HP MPower

X stations support all HP MPower functionality with the exception of a live video input. X stations are not true HP MPower clients. They require that both the HP MPower server and client run on the host. The newly introduced HP ENVIZEX stations support local audio, local scanner, local floppy diskette in DOS format, and HP SharedX functionality as a sender and receiver. The following additions are recommended to the .vueprofile file in a home directory for a user that uses an X station and the HP MPower software.

```
# AUDIO and SCANNER variables are derived from DISPLAY variable to ensure
# that multiple X stations can support local audio and local
# scanning while running HP MPower software on the same host.

xterm_name=$(echo $DISPLAY | sed 's;:.*;;')

SCANNER=$xterm_name
export SCANNER

AUDIO=$xterm_name:0
export AUDIO

#SPEAKER=INTERNAL        # uncomment if there are no external speakers
SPEAKER=EXTERNAL         # comment out if there are no external speakers
export SPEAKER
```

Initially, our strategy was to support as small a number of configurations as possible and to increase that number with each subsequent product release. We began by providing support for the Series 700 only. With HP MPower 1.2 we expanded that base to include the Series 800 as HP MPower servers, the Series 300 and 400 as HP MPower X stations, and the new ENVIZEX X stations as media-enabled X stations. Similarly, our first release was English only; HP MPower 1.2 added support for Japanese and 16-bit character sets.

The team provided an alpha release to a select number of customers, which was hand-delivered and installed by a support group from the factory. We also created two separate beta releases to flush out installation and configuration problems. More than 40 internal HP sites installed early versions of the software. Our goal was to minimize the amount of time required to install more than 30M bytes of software.

Two usability tests helped shape the user interface of the system. We learned early that the user interfaces of the individual components had to change to fit the overall user interaction with the system. For example, all components adopted a uniform file selection dialog to enable users to access and save files consistently.

A system administration walkthrough resulted in installation and documentation adjustments, particularly in organizing the system installation into separate procedures for instant ignition and noninstant ignition systems. The prerelease feedback enabled us to cut the installation time from two weeks (at the project outset) to two days (at alpha release) to two hours (for the final product).

We divided testing responsibility among the component owners so that one team tested the fax, imaging, audio, and other media components, another team tested the HP SharedX and whiteboard components, and a third team covered the mail, desktop integration, and system installation areas. We relied extensively on an automated defect tracking system for monitoring defect levels and resolution rates on a weekly basis.

We performed a limited set of code inspections on new and critical modules. Team members spent many hours testing the components and system interactions manually. They have since started work to automate a number of these tests.

Diagnostics

HP MPower has a diagnostic facility called Dr_MPower which consists of a variety of submodules that check dozens of key system and personal files to ensure that the system and the media services are properly installed and configured. Dr_MPower can be run on either a server or a client.

Users can run Dr_MPower simply by double-clicking the tool's icon in the file manager toolbox. See the article above for more about this diagnostic.

Challenges

Besides working on a tight schedule, some of the challenges we encountered while developing the first HP MPower product included:

(continued on page 18)

The HP Instant Ignition Program

The HP Instant Ignition program is focused on increasing customer satisfaction by delivering a complete, integrated, preconfigured system† that is ready to use. The program has the following goals:

- Give the customer a positive "out of the box" experience (i.e., a positive first impression of our system)
- Decrease the "time to productive use" of a system
- Decrease HP's field and factory cost by using common tools and standardized processes.

What Is an Instant Ignition System?

The HP Instant Ignition system is made up of the basic computer system hardware, including a disk that is preloaded with an operating system, optional application software, and system documentation.

The HP Instant Ignition design team set out to make HP 9000 systems less intimidating to the user. The team designed a number of usability enhancements aimed at addressing the program goals. For example, esoteric messages were removed from the boot process. The team replaced other messages with a concise checklist that indicates passing or failing portions of the boot procedure (Fig. 1). Labels on boxes were enhanced to make them more readable.

Additionally, the first time the system is booted, the user is prompted to provide system parameters that cannot be predicted in the factory, such as networking specifics. This method of configuring the system eliminates the need for a user to edit files manually.

The HP MPower team designed an extension to these system parameter prompts. The links between the client and server are established as the system is booted so that HP MPower is fully functional the first time any user logs into the system.

CHAMP

To preload the various operating systems and applications, the instant ignition team developed a software tool called CHAMP (channel-reseller and manufacturing process). CHAMP fits into an automated hardware manufacturing line at a point where the CPU is assembled with a disk and diagnostics have successfully completed. At this point, the customer's disk is ready to be built using a two-phase process.

First, the customer's CPU boots from a dedicated manufacturing disk that contains the CHAMP tool. CHAMP runs on the customer's hardware and downloads the operating system to the customer's disk or target disk. CHAMP then downloads some utilities to the target disk to prepare for phase two. The last thing to occur in phase one is to reboot the customer's system.

In phase two, the customer's system boots from its own disk, the target disk created above. If necessary, instead of booting to the login screen as expected, the system runs the utilities downloaded in phase one to fetch additional software. This software may come from a `netdist` server or from another system on the network. The system will also configure its kernel, if necessary, based on the customer's hardware and optional software.

The last thing to occur during a CHAMP build is to remove the CHAMP utilities and return the operating system to a pristine state. The system then shuts itself down and is ready for the customer.

Design Considerations

In designing CHAMP, a number of requirements had to be met. Some of the key design requirements were to:

- Preload HP 9000 Series 700 and Series 800 HP-UX operating systems
- Run from a command line, allowing existing manufacturing processes to invoke CHAMP automatically
- Run on the customer's hardware (This guarantees that the HP-UX kernel and the recovery instructions are specific to the customer's hardware.)
- Load application software that is delivered in a variety of formats (HP's preferred method for packaging software is update format. The utility `/etc/update` is a standard part of the HP-UX operating system and is available to internal and external customers. CHAMP is flexible enough to load software in other formats, such as `tar` and `cpio`.)
- Load software in sequence (This means that if application 1 must be loaded before application 2, CHAMP can accommodate this sequence.)
- Generate customized recovery instructions for each system. (These instructions outline exactly how the system was created in manufacturing so that the customer can recreate the original system in the event of a disk failure. These instructions are online in a file, and a hard-copy version is added to the shipping boxes at the final stage of the manufacturing process.)

Other Uses for CHAMP

It became evident that CHAMP had potential uses outside of HP's manufacturing process. For instance, resellers could be more effective in delivering HP Instant Ignition systems if they had access to CHAMP and if CHAMP was easy to use and maintain.

While flexibility is one of the outstanding features of the CHAMP tool, flexibility does not always equate to ease of use. To enhance usability, a graphical user interface was layered onto the basic CHAMP tool. With this interface, CHAMP is very easy for the nonexpert, such as a reseller, to use to build preloaded systems quickly.

CHAMP is also used by operating system and application developers internal to HP. These developers can use CHAMP to build a test system for their development. It takes about 30 minutes to create a new HP-UX system using CHAMP. This is considerably faster than using CD-ROM media to build a new system. All HP teams that develop pieces of the HP-UX operating systems and preloaded applications are required to test their software on the base system built by CHAMP.

Conclusion

Customer acceptance and popularity of the HP Instant Ignition program continues to grow. Customers like to receive their systems with preloaded software. Today the HP Instant Ignition program is limited to preloading HP applications because of internal issues related to selling and preloading third-party software.

Sue Magenis
Development Engineer
Open Systems Software Division

† Currently HP 9000 Series 700 and 800 machines.

HP-UX Start-up In Process	Status
Initializing System	[OK]
Starting Networking	[OK]
Starting Diskless Cluster Client	[OK]
Starting System Functions	[OK]
Starting Auditing	[OK]
Starting Diagnostics	[OK]

(a)

HP-UX Start-up In Process	Status
Initializing System	[OK]
Starting Networking	[FAIL]*
Starting Diskless Cluster Client	[OK]
Starting System Functions	[OK]
Starting Auditing	[OK]
Starting Diagnostics	[OK]

NOTE: An error has occurred!
Refer to the file /usr/adm/rc.log for more information.
Press [Return] to continue...

(b)

Fig. 1. An HP Instant Ignition boot checklist. (a) When things are OK. (b) When things are not OK.

Diagnosing and Reporting Problems in the Multimedia Environment

The need for installation and configuration diagnostics for user interface software became apparent soon after the release of HP VUE 2.0. Because of the large number of files associated with HP VUE and the dissemination of those files across the file system, response center† engineers were spending a lot of time telling customers about which files to check for known problems. An analysis of call-in data from the Atlanta response center revealed that the amount of time spent each quarter handling HP VUE configuration questions was rising rapidly.

As a result of this feedback, the technical training and support group in Corvallis decided that a script or program could more efficiently wander the highways of the file system and check on the existence, permissions, and ownership of most of the files that HP VUE depended on. Key files, such as /usr/adm/inetd.sec, /etc/inittab, and others could be searched with standard tools to determine if the attributes required for certain entries were proper. The use of a script could reduce the amount of time required to check basic configuration issues from hours or days to a matter of a minutes. Thus, Dr_VUE (diagnose and report Visual User Environment problems) was born.

Dr_MPower is a direct outgrowth of Dr_VUE. Dr_MPower consists of a collection of diagnostic tools used to help debug installation and configuration problems in the HP MPower environment. Since HP MPower combines several disparate applications, we decided to create a separate script to check each individual component. This resulted in a total of 12 scripts: one each for 10 components of HP MPower, a file that contains common functions used by each of the scripts, and a calling script. The calling script is Dr_mpower, which does some checking of system functions and then calls each of the remaining scripts in turn. Since HP MPower makes heavy use of the HP VUE environment, an action was defined to initiate Dr_MPower. This action is located in one of the toolboxes available to all users. A simple double-click on the Dr_MPower icon results in a new terminal window that displays the information output by Dr_MPower. Each of the scripts called by Dr_MPower can also be run individually to perform checking on the separate components of the HP MPower environment.

A myriad of configuration issues exist in the HP MPower environment. Dr_MPower certainly does not check each and every possible one, but rather looks at what we hope to be the vast majority of them. Since the MPower environment consists of servers, maxiclients (clients running HP VUE locally), miniclients (clients running HP VUE on the server), and X terminals, the first thing that Dr_MPower needs to determine is the type of system it is running on. When this has been accomplished, Dr_MPower will check items specific to that environment. For example, there are differences between the client and the server in recommended kernel parameters, as well as certain processes that run on the server and not on the client. Another difference is the fact that the miniclient does not run HP VUE locally, so the check_vue script that is called by Dr_MPower should not be run for that platform.

The installation and configuration of HP MPower makes heavy use of scripts, and since scripts fail occasionally, Dr_MPower attempts to verify that all the actions performed in the various configuration scripts have been accomplished. An example is checking in the file /etc/netnfsrc to see if the parameters NFS_CLIENT, NFS_SERVER, and START_MOUNTD were successfully set to one by the configuration scripts. Another example is checking to see that a line was added in /etc/rc to start the font server and that the font server is currently running. Fig. 1 is a graphical representation of the components checked by Dr_MPower.

Anytime Dr_MPower encounters something that appears to be in error, a message is written that attempts to inform the user as to the severity of the problem with either INFO, WARNING, or ERROR statements (similar to the feedback seen in an update log). If possible, an appropriate course of action is also given. For example, a check is made of the /etc/src.sh file to determine if the configuration script added two entries specifying the HP MPower and HP VUE servers. The following code performs this check and issues a WARNING statement if the installation script did not add the entries in the /etc/src.sh file:

```
## look in /etc/src.sh to see if VUE_SERVER and MPOWER_SERVER were
## added

count=$(grep -e VUE_SERVER -e MPOWER_SERVER /etc/src.sh | wc -l)

    [ $count -eq 2 ] ||

    print "WARNING:The MPower installation script should have added
      two entries to /etc/src.sh:
      VUE_SERVER=$hostname; export VUE_SERVER

      MPOWER_SERVER=$hostname; export MPOWER_SERVER

      This does not appear to be the case. You should rerun the
      /usr/MPowerServer/setup_server script or add these
      lines yourself."
```

This code performs the check and provides the user with two options to correct the perceived problem: either manually add the entries or rerun the configuration script that should have made the additions.

There are over 4000 lines of code in the various scripts that make up Dr_MPower. Considering the positive feedback we have received from support partners concerning Dr_VUE, we are confident that the Dr_MPower scripts will significantly reduce the time spent supporting customers with HP MPower problems.

John V. Peterson
Support Engineer
Workstation Group/Corvallis

† HP has several locations worldwide that are responsible for handling software or hardware problems from customers who have purchased response-line support contracts. These locations are called response centers.

- Geographically separated teams. The HP MPower project team included members from Oregon, Massachusetts, California, Colorado, and Canada. Keeping the lines of communication open and efficient helped us understand some of the requirements of distributed work groups.
- Integrating disparate components. Initially, team members designed their products (such as fax and whiteboard) as standalone applications. Achieving a common look and feel involved changing parts of our HP MPower subsystems such as interprocess communication mechanisms, icon visuals, online help, and dialog box behavior.
- Incorporating a common file-typing mechanism. The HP-UX file system is designed to treat all files simply as bags of bytes. Within a graphical user environment, it is necessary to know file types to determine the appropriate actions for a given object (e.g., mail, print, edit, fax, play, and view). To preserve the file types of HP MPower media objects, we implemented a variety of file-typing mechanisms, ranging from appending file name suffixes to inspecting file contents.
- Living within workstation resources. OSF/Motif applications are quite large. Thus, running a variety of OSF/Motif-based applications and a myriad of server and helper processes on limited-memory systems forced us to adopt a client-server architecture.
- Large media objects. Media objects are by nature large. For example, a sound clip can cost 16K bytes per second, and each video frame can cost almost a megabyte. We use a number of compressed file formats to keep file sizes to a minimum such as the JPEG file format for video images. Fortunately, the processing power of the HP PA-RISC-based workstations makes the compression and decompression of media objects a fairly painless process.
- Configuring the distributed environment. System administration burdens can grow exponentially when services are

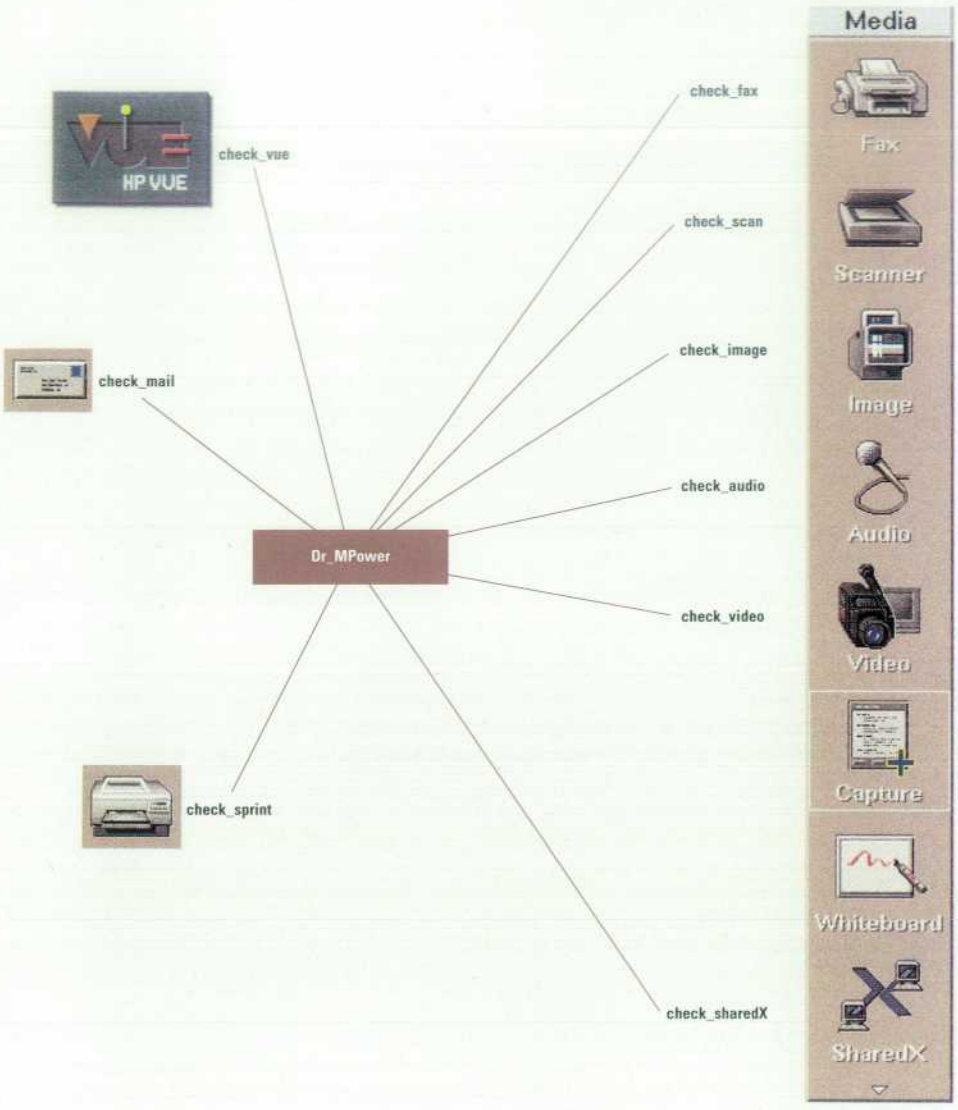


Fig. 1. The HP MPower components checked by Dr_MPower and the associated scripts responsible for making the checks.

distributed over several hosts. By offering a fairly restricted default configuration, we have reduced the headache of installing and maintaining HP MPower nodes.

- Living in the network home. Today's users are familiar with desktop computing, in which most of their applications and data reside on their local machine. When their environment is distributed across multiple hosts, users can become disoriented in the network environment. We have tried to make file access as transparent as possible and to invoke applications on the expected host.

Acknowledgments

The HP MPower team would like to thank the instant ignition team at Fort Collins, Colorado and the Exeter Computer Manufacturing Operation team at Chelmsford, Massachusetts for their help in delivering this fast-track product to market. Thanks to Teri Wilson, the Corvallis manufacturing coordinator, for expediting the release and shipping process. The user interface design team, headed by Barry Mathis, provided invaluable help in desktop visual integration. Thanks to Andreas Scheel, Larry Rowland, and Tim Yuen for their evaluations of the HP MPower client/server architecture and system administration. Finally, a special thanks to Takao Miyake for localizing the HP MPower product for Japan.

A Graphical User Interface for a Multimedia Environment

The HP Visual User Environment, or HP VUE, provides not only a friendly user interface to the HP-UX* operating system but also a framework for the HP MPower system.

by Charles V. Fernandez

It was inevitable that once the multitasking, multiuser, and network capabilities of the UNIX* operating system were connected with the power of graphics workstations that the next step would be to civilize the UNIX command line interface with a graphical user interface (GUI). Graphical user interfaces are literally changing the face of UNIX systems, and in doing so, are helping to spread UNIX systems and workstations from their historical installed base among technical users into the broader markets offered by commercial computing.

Among these GUIs is the HP Visual User Environment (HP VUE). HP VUE is the first GUI to provide the following features and capabilities for workstations running the HP-UX operating system:

- PC-compatible controls
- 3D visual appearance
- A graphical user interface to the system's particular functionalities while hiding the peculiarities of the system from the end user
- Multiple levels of integration for in-house and ISV (independent software vendor) applications.

As the framework for HP MPower, HP VUE provides the structure into which multimedia components can be integrated.

HP VUE provides a consistent set of controls with which to operate a workstation. While UNIX system commands contain a lot of functionality, this functionality is often cryptic, hard to understand, and difficult to remember, especially for occasional users. Additionally, UNIX system commands and their options are often inconsistent (some commands provide output, others don't, and what an `-o` option means depends on the command, not the functionality of the option). HP VUE changes all this. HP VUE uses a simple set of graphical controls, consistent with the Common User Access (CUA) model followed by Microsoft,® IBM Corp., and many other PC manufacturers. The CUA model is based on pushbuttons, scroll bars, and menus.

A user familiar with similar controls such as Microsoft Windows can sit down at a workstation with an HP VUE user interface and immediately take control because the skills required to operate a PC GUI are the same as the skills needed to operate an HP VUE workstation.

To operate a UNIX system from the command line, a user has to type commands and command options at the keyboard. A spelling mistake or a typing error could mean disaster. Commands, unless memorized, have to be looked up in the documentation. HP VUE unburdens the user from having to memorize UNIX commands and retype faulty command lines. To operate an HP-UX system from HP VUE, a user directly manipulates the graphical objects that populate the workspace. For example, to move a file from one directory to another, the user drags the file icon from one file manager view to another and drops it there. To start an application, the HP VUE user double-clicks the application icon.

One of the confusing things for users of an operating system is that they are required to develop a three-tiered level of consciousness: one level for the operating system, one for the application, and a third for their data, the only tier they are really interested in. New users have a difficult time distinguishing where one level stops and the other starts. Command line environments routinely demand that a user who wants to access data must switch from a data focus, remember which application works with that data (and possibly where that application is located), and negotiate how to start the application. Only after successfully starting the application can the user return to the desired data focus.

HP VUE, on the other hand, because it associates applications with data using an action and file-typing function, enables users to remain focused on their data, and the operating system and application tiers remain hidden by the user interface. To access data in the HP VUE environment, users double-click the data icon. The application starts automatically and loads the selected data file, leaving the user free to focus on work, not the mechanics of getting to work.

The HP VUE 3.0 Design Process

Getting HP VUE to its current state has been an evolutionary process. The process began in the mid-1980s when HP adopted the X Window System as the strategic graphical layer for workstations. This evolution continued through the development of the 3D window manager, `hpwm`, its submittal and acceptance by the Open Software Foundation (OSF) as an industry standard, the proliferation of OSF/Motif, the development of HP VUE 2.0, and finally, HP VUE 3.0.

During the design process of HP VUE 2.0 it became apparent that designing a user interface without user input would be a recipe for disaster. For the development of HP VUE 3.0, a more formalized approach to user input was established. This approach goes by the acronym QFD, for Quality Function

Deployment.[1] It was adopted for the development of HP VUE 3.0 to help ensure that:

- Customer input was systematically collected
- Customer input was quantified to define and prioritize product requirements
- Customer input was factored into the design process
- Product design was affected by the input as opposed to the input being interpreted to fit the design.

The QFD for HP VUE 3.0 was a multistep process. Keeping in mind that the primary target market for HP VUE consists of workstation users, we established a target design market for HP VUE 3.0 in the areas of factory floor operations, scientific research, industrial and architectural design, information engineering (knowledge workers), design engineering, education, CASE (computer-aided software engineering), and system administration. These design markets were given relative weights to facilitate the prioritization of their inputs. For example, input from scientific research with a weight of 5% wasn't nearly as influential as input from a knowledge worker with a weight of 35%. Within these areas, potential users were categorized on a UNIX knowledgeability spectrum that included categories for protected users, naive users, moderate users, and sophisticated users. Correlations were noted between these user types and our target design market. Over 30 companies were visited and asked to input into the HP VUE 3.0 QFD process.

The result was a weighted list in priority order of what customers wanted. Performance figured high on the list. Pizzazz† was also important. Of less importance were multiple fonts and workspace manager button labels. This list essentially formed a prioritized functional specification wish list for the HP VUE 3.0 product.

The Workspace Manager

HP VUE 3.0 workspace manager has a new look that differentiates it from HP VUE 2.0. This new look is part pizzazz and part pragmatism. The pizzazz is that the square button look of the 2.0 front panel is replaced by a "membrane" look in which the bevels demarking the buttons are removed and the icons that form the button labels appear inset into the front-panel membrane.

Fig. 1 illustrates the familiar "boxy" look of the HP VUE 2.0 front panel and the new and improved membrane look of the HP VUE 3.0 front panel. Notice also the slide-up subpanels available with the HP VUE 3.0 front panel. The tall subpanel on the left is the HP MPower media panel. HP VUE 3.0, as mentioned earlier, provides the framework for HP MPower.

Aside from supplying an easily recognizable visual distinction between the two versions of HP VUE, the membrane look makes the front panel easier to configure. To place a button in the old front panel required users to count the length and width of a button in pixels and then add pixels for the bevels. This proved to be a time-consuming and error-prone activity and certainly not user-friendly. The membrane look eliminates the need for users to count pixels. They simply specify the button, and the front panel magically grows itself to fit the new button.

† Pizzazz refers to those features in a product that might create customer excitement (e.g., garbage can icons that open when trash is tossed in or button icons that look and behave like real pushbuttons).

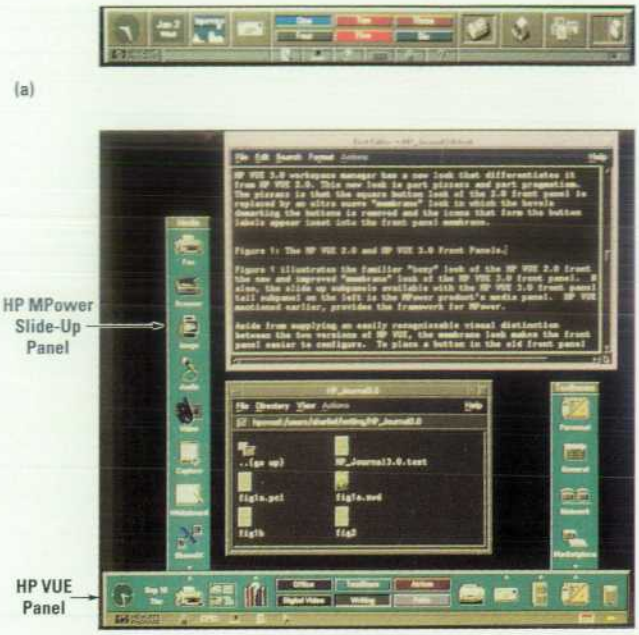


Fig. 1. (a) The "boxy" look of the HP VUE 2.0 front panel and (b) the new and improved membrane look of the HP VUE 3.0 front panel including the HP MPower slide-up subpanel on the left.

Another pizzazz and functional feature of the new front panel is the inclusion of slide-up subpanels. These panels slide up seemingly from behind the front panel when their control is pressed. The slide-up subpanels enable developers to make more controls readily available without taking up more space. They also provide a convenient place to locate the multimedia components of HP MPower. The slide-up subpanels can be torn off and posted to the workspace like a control panel.

The File Manager

The file manager is a good example of how QFD and usability studies can refine a product. HP VUE 2.0 packed a lot of functionality into the file manager. Virtually all system file management was available through the file manager's GUI. This is a good thing. However, what the 2.0 file manager didn't do, and what QFD and usability studies made apparent, was that all that functionality wasn't readily accessible.

The best example of this is the process of renaming a file. To rename a file in HP VUE 2.0, the user had to do the following:

- Select the file to be renamed
- Pull down the File menu
- Select Rename from the menu
- Wait for the Rename dialog box to appear
- Click the Rename text entry field in the dialog box
- Type the new file name
- Click the OK button.

As a result of feedback from QFD and the usability studies, the HP VUE 3.0 file renaming process involves the following steps:

- Click on the file name to change
- Type the new name
- Press the **Return** key.

Visually, the file manager didn't change that much between HP VUE 2.0 and HP VUE 3.0. However, the difference in ease of use and accessibility to file management functionality is quite significant.

The Style Manager

The HP VUE 3.0 style manager is another component that shows the subtle but unmistakable signs of the QFD process. While the work environment at HP is fairly open and users are fairly knowledgeable about the UNIX operating system, some of the system administrators and managers from customer sites often had more controlled environments for security reasons. Their feedback was the impetus for removing the Host dialog from the style manager. They wanted control over the X Window System's ability to host "foreign" workstations. Also, for security reasons, they wanted to control the ability of one workstation to host other workstations.

Along the same lines, security-conscious QFD participants requested that the little lock icon that appears on a locked workspace be changed to a full screen cover so that a passerby could not read information visible in the windows left open on the workspace.

Performance

Talk about graphical user interfaces long enough and eventually the discussion will get around to performance. Since HP VUE 2.0 is the most visible part of the operating system, it bore the brunt of a lot of negative performance comments.

Some of these criticisms, like "HP VUE is a big memory consumer because it takes up 16M bytes of RAM," are undeserved. HP VUE isn't really a heavy memory user, but to many users it may seem to be so. On a typical workstation with 16M bytes of RAM, the RAM is apportioned roughly as follows:

Miscellaneous Daemons	3M bytes
File Buffers	3M bytes
Kernel	3M bytes
X Server	2M bytes
Workspace Manager	1M byte
File Manager	0.75M byte
Help Manager	0.75M byte
Style Manager	1M byte
Hpterm Console	0.75M byte
Message Server	0.5M byte
Total	15.75M bytes

Nearly three quarters of the 16M bytes HP VUE is supposedly hoarding is actually being used by core system functions. The point that is most often misunderstood about HP VUE is that it is not a monolithic application, but a set of six components, not all of which need to be running at the same time. When something isn't running or being currently used, it is pushed out of RAM. As a matter of fact, at a minimal level, the user can run the workspace manager and receive the benefits of multiple workspaces for little more cost in RAM than would be experienced with using a standard OSF/Motif window manager.

One of the criticisms that HP VUE does deserve is for the amount of time it takes to log in. For HP VUE 3.0, performance, as mentioned above, was a key design area. Besides speeding up the access to functionality like renaming files, which increased perceived performance, all HP VUE components and major processes were studied, including login, logout, file management, and session management. Customers were asked what trade-offs in functionality would be acceptable for the sake of better performance. Out of this research came a number of improvements. Instead of starting all HP VUE components immediately at login, their individual starts are staggered. Studies showed that starting processes all at once caused tremendous contention for RAM, while actually delaying a component's start until the preceding component was fully started. This staggered start of all processes reduced overall startup time.

Another result of the HP VUE 3.0 performance work was the development of a lighter-weight version of HP VUE. This lighter version of HP VUE has two of the most RAM-expensive components removed: session and file management. Reliance on a default HP VUE session instead of true session management speeds up the login and logout processes. Using the file management functions available on the file menus of all standard OSF/Motif applications instead of the HP VUE file manager avoids session slowdowns caused by the file manager's periodically jumping into RAM (and pushing the resident application out) to check that its file views match the current file structure.

Conclusion

One of the most gratifying results of the HP VUE 3.0 QFD effort was the affirmation from customers of our belief that a GUI is a work in progress, an evolutionary process. Dramatic differences between HP VUE 2.0 and HP VUE 3.0 were both uncalled for and unwanted. What customers did want was evolutionary changes, such as to make it easier to rename a file and configure the front panel. Our QFD customers were very attuned to HP VUE as an environment from which to access their working applications. Understanding their vision enabled us to take the next step in HP VUE's evolution: making HP VUE the framework for HP MPower.

Reference

1. S. Graves, W. Carmichael, D. Daetz, and E. Wilson, "Improving the Product Development Process," *Hewlett-Packard Journal*, Vol. 24, no. 3, June 1991, pp. 71-76.

HP SharedX: A Tool for Real-Time Collaboration

With this real-time communication product, two or more remote users can share and interact with the same X-protocol-based applications from their workstations. Windows are shared in such way that it almost seems as if all the participants in the shared session are sitting at the same workstation, running the same application.

by Daniel Garfinkel, Bruce C. Welti, and Thomas W. Yip

HP SharedX is a communication tool that extends the industry-standard X Window System to enable real-time sharing of X-protocol-based applications between two or more remote users and displays. With HP SharedX users can share information with one another via a workstation without being in the same location.

Being able to share information over a network in real time is a very effective productivity tool. The following two examples show how displaying applications across a computer network can increase productivity:

- System administration. Mary is a system administrator for several networks distributed over several widely dispersed buildings. When a user on a particular network encounters what may be an application or system problem, Mary can, with the user's cooperation, establish a common (HP SharedX) window with the user's system. With this shared window Mary can see and diagnose the user's problem while the user is watching what she is doing. She can perform this task without leaving her desk, using the diagnostic tools available at her workstation.
- Remote demonstrations. For some new or complex software packages, there may be only a handful of people who are able to demonstrate the software effectively. HP SharedX gives these people an opportunity to have a virtual presence at remote locations on the network. Hewlett-Packard's Work Management Operation routinely uses HP SharedX to demonstrate advanced features of its HP WorkManager database product from the desks of engineers in Fort Collins, Colorado to prospective buyers all over the world. Also, various partners who are developing additions to WorkManager use HP SharedX to demonstrate their work in progress to the engineers in Fort Collins, allowing new features to be evaluated "live" without travel.

HP SharedX can accomplish display sharing because of the nature of the system it is built upon, the X Window System.[1,2] The X Window System, known simply as X, is a networked window system allowing X applications running on one computer to display on another (see "X Window System Client/Server Architecture" on page 25). This networked aspect allows sharing of expensive high-speed computers among several users on low-cost X-based display stations. Furthermore, X is designed to be interoperable in heterogeneous computing environments. Applications running on one vendor's hardware can display themselves on another vendor's display by adhering to the X protocol standard. Heterogeneity is accommodated by abstracting the application's view of the display subsystem, including the graphics hardware, input device control, memory management, and data formats. Interaction with the display station occurs through a well-defined protocol, which provides a consistent way for applications to work with a wide variety of display devices, from PCs to high-resolution, deep-color workstations.

X has become the industry standard window system for workstations in part because of its networked and heterogeneous nature. HP SharedX builds on the X Window System by retransmitting the X protocol stream to multiple X display stations, thus simultaneously displaying a single application on multiple computer displays. Some key features of HP SharedX are:

- No changes are needed to existing X applications to share them. HP SharedX allows users to share virtually any X application, rather than forcing users to use specially written, collaborative applications.
- Users see a copy of the application to the best ability of their display device. If necessary, HP SharedX will degrade the quality of the displayed images to match the capabilities of the display hardware.
- Running applications can be shared without prior setup. Users need not restart applications they want to share; the application can be shared in its current state. This is important in consulting environments where users may not know how they got their application into its current state.
- The receiving machines (machines receiving the shared application) need no modifications or special software installed. Since X protocol is the communication mechanism, the receiving machine can be any X-capable display device.
- Receivers of a shared application can interact with it as well as the sender (the machine sending the shared application). Receivers can demonstrate operation of the application rather than describing to the sender what to do.

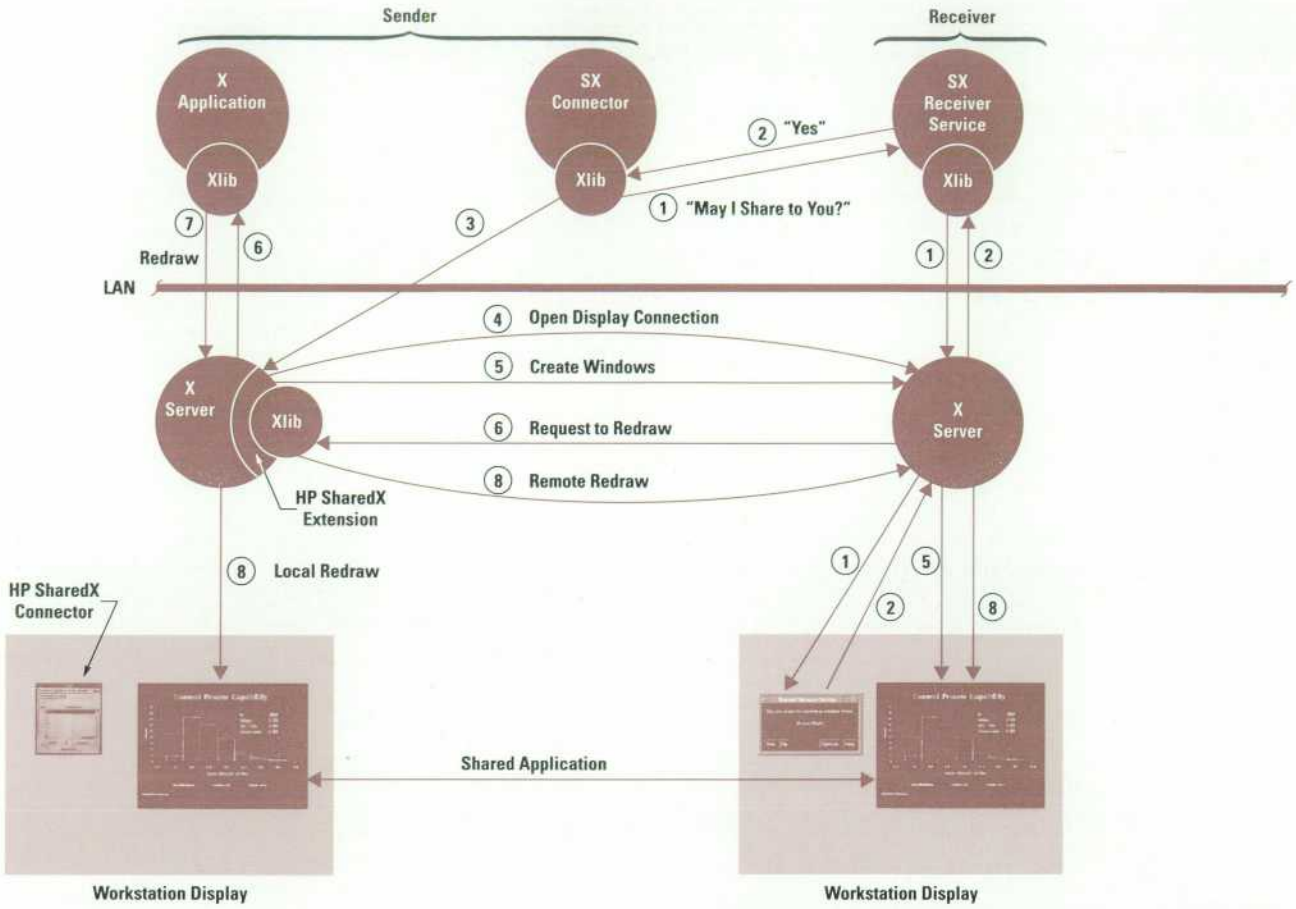


Fig. 1. HP SharedX sender/receiver architecture and data flows during initial display connection.

The rest of this article covers the architecture of the HP SharedX system, the user model and user interface for sharing, an overview of the low-level mechanism that multicasts the X protocol and merges input events, and finally, a description of how sharing is accomplished with displays of different visual types and different resources.

HP SharedX Architecture

HP SharedX consists of three components: the HP SharedX user interface (known as the connector), the HP SharedX receiver service, and the HP SharedX extension to the X server (see Fig. 1). The connector is the sender's control for sharing and unsharing windows, enabling and disabling receiver input, displaying sharing status, and so on. The receiver service is an optional process on the receiver's machine that simplifies sharing windows and increases security. The HP SharedX extension is a low-level mechanism that shares windows, keeps those windows up to date, and merges input from multiple users. These three components are described in more detail later.

An HP SharedX session begins when the user at the sending workstation specifies a receiver and pushes the Share button in the HP SharedX connector window (Fig. 2). After the Share button is pushed the sender selects the window to share by clicking the mouse button over the desired window. Once the desired window is selected the following events occur between HP SharedX processes on the sender and receiver workstations shown in Fig. 1. Each step corresponds to a number circled in Fig. 1.

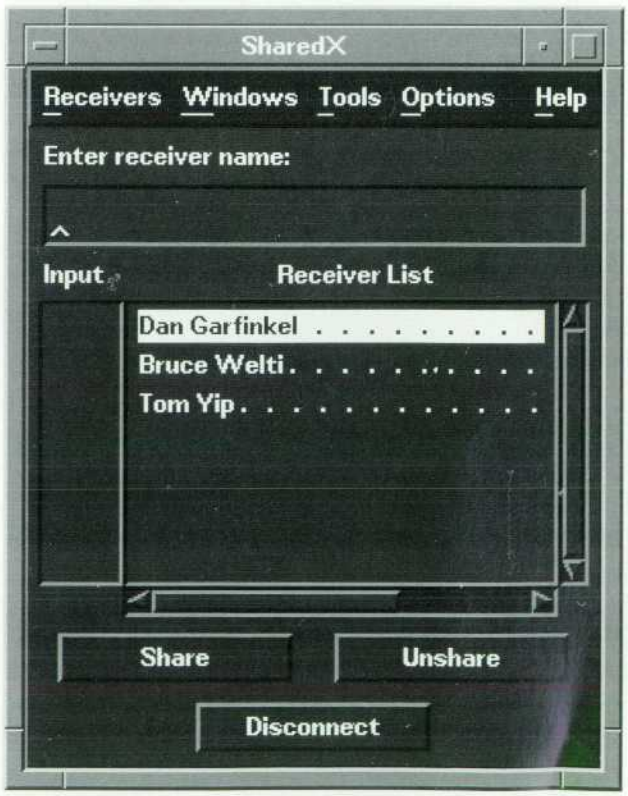


Fig. 2. The HP SharedX connector window.

X Window System Client/Server Architecture

The X Window System, commonly referred to as X, is an industry-standard, network-transparent window system. X presents to the user a hierarchy of resizable overlapping windows providing device independent graphics. A graphical user interface is commonly included as an integral part of the X Window System.

The principal feature that distinguishes X from a conventional window system is its network transparency. The X Window System allows window applications or clients to access the display only through the X server, which is a separate process that arbitrates resource conflicts and provides display, keyboard, and mouse services to all clients accessing the display (Fig. 1). X can support a spectrum of hardware displays ranging from small monochrome units to advanced graphics systems with up to 32 bits of color per pixel.

The client and the server exchange information only by means of the X Window System protocol which can be implemented via any reliable byte stream. In the HP-UX* implementation of X, as in most others, this byte stream is implemented as a socket, which is a logical data connection between two processes on the network. Clients may reside locally with the display server or on a remote system across the local area network (LAN). A performance optimization bypasses the physical LAN when the client and the server are on the same computer.

Because the client program and the display server are two separate entities, the target display can be specified at the time the application client is run. The client program is indifferent. It sends out X protocol commands via calls to the X library (Xlib), which in turn calls the network service functions to communicate with the target X server.

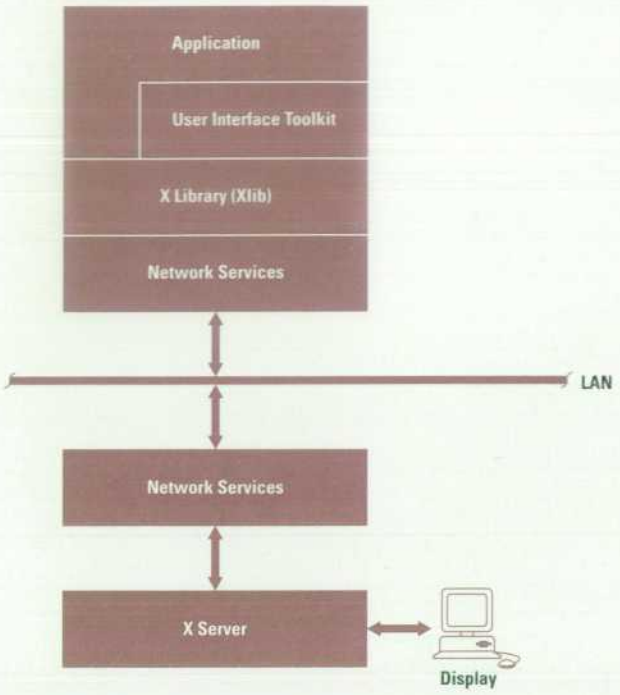


Fig. 1. X Window System client/server architecture.

1. The connector sends a message over the network to the HP SharedX receiver service on the receiver's machine. The receiver service displays a window asking the user at the receiver system to accept a shared window from the sender (Fig. 3).

2. The receiver passes back a yes or no response to the connector. If the answer is no, the sender is informed that access is denied to the receiver's machine.

3. If the answer is yes (or if the receiving machine does not have the receiver service), the connector sends a special X protocol request (share window W to receiver R2) to the sender's X server. The X server's main event loop recognizes the share window request as one directed at the HP SharedX extension. It calls the appropriate function in the extension to handle the request.

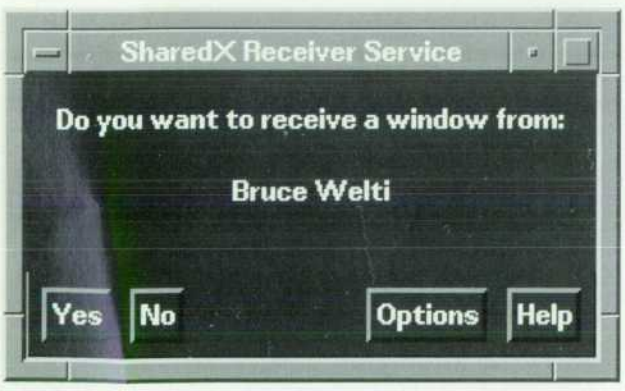


Fig. 3. The HP SharedX receiver service window.

4. The HP SharedX extension opens a connection over the network to the receiver's X server. This is done with the Xlib call XOpenDisplay.

5. The HP SharedX extension creates windows on the receiver's screen to match those on the sender's screen.

6. As the windows are created and mapped, the receiver's X server generates a request to redraw the newly created windows.

7. The application responds by redrawing the contents of all of its windows. X protocol drawing requests to redraw the windows are handled locally by the sender's X server, as usual, but since the application is now shared, the protocol is effectively echoed to the receiver (⑧ in Fig. 1). This is how the receiver is brought up to date with the sender when the share is initiated. Incidentally, none of the resources (graphics contexts, fonts, and so on) exist on the receiver when the requests start flowing from the shared application. Resources on the receiver are created when they are needed, that is, at the point they are first used, and not when they are created by the shared application. As a result, the input from the application to the sender's X server and output from the sender's X server to the receiver's X server might look like the following streams:

Input	Output
ClearWindow(win1)	ClearWindow(win1a)
DrawLine(win1, gc1, x, y)	gc1a = CreateGC(win1a, ...)
DrawLine(win2, gc2, x, y)	DrawLine(win1a, gc1a, x, y)
	gc2a = CreateGC(win2a, ...)
	DrawLine(win2a, x, y)

Graphics Glossary

The following are some graphics terms that may be of help in reading parts of this article that discuss graphics.

Color Map. A very high-speed look-up table that maps the numbers stored in the frame buffer (video memory) to actual color values which are converted to analog voltages and sent to the monitor.

Frame Buffer. The video memory of a display device in which each element represents one picture element, or pixel. The frame buffer is divided into two parts: onscreen memory (current visible image) and offscreen memory (graphics memory that is never visible).

Graphic Context. A set of attributes such as foreground and background colors, line styles, and fill patterns which are used by X clients to specify how the X server should render the drawing requests it receives.

Image Planes. The primary display memory on HP's display systems, used for rendering complex images.

Offscreen Memory. A portion of the frame buffer that cannot be displayed on the monitor. In all other respects, offscreen memory behaves the same as on-screen (visible) memory. Starbase and X use offscreen memory to hold character, cursor, pixmap, and scratch information for rapid transfers to onscreen memory.

Pixmap. A rectangle of image data maintained in offscreen memory when there is room, and in virtual memory when there is no room in offscreen memory.

Rendering. Any form of drawing operation, including text, line, and raster output. Rendering may occur to onscreen memory, offscreen memory, or virtual memory.

Visual Type. The color map capabilities of a given display. Common visual types supported on HP displays include 1-bit static gray (or monochrome), 8-bit pseudo color (having 256 color map cells of RGB values), and 24-bit direct color (using 8 bits each for red, green, and blue values).

Note that these are only representations of the drawing commands sent from the application and the sender's X server. The parameters win1a and win2a correspond to the different windows created in step 5. The identifiers win1a, gc2a, and so on represent resources on the sender's display which are mapped to resources win1a, gc2a, and so forth on the receiver's display.

Once the redraw step is finished, the share connection is established. From here on, the two displays are kept in synchronization by equivalent X protocol being sent to each server.

Connector

The HP SharedX connector (Fig. 2) is an X Window application that contains all the controls that enable users to set up and control application sharing. Using task analysis, we designed the HP SharedX connector to match the user's task flow, thus providing intuitive controls to the complex operation of sharing an application. Some of the usability features of the connector window include:

- An address book (Fig. 4), accessible from the Receivers menu item in the connector window, to keep and organize common receivers and allow users to specify receivers by user name rather than machine name
- A shared pointer (called a telepointer), available from the Windows menu item in Fig. 2, for users to point to areas of shared applications during collaboration

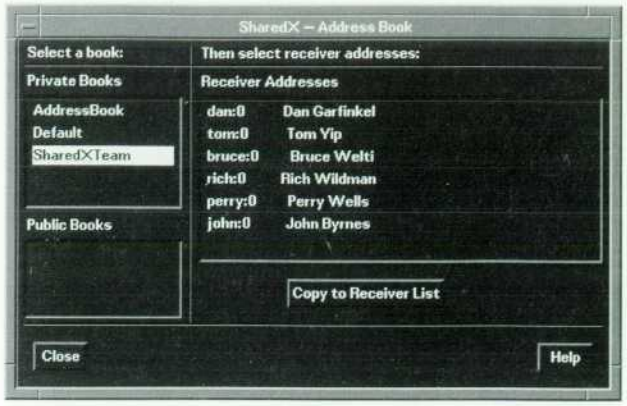


Fig. 4. A window showing the address book.

- Several cues, like changing a window title so that the user can keep track of the windows being shared, the users receiving shared windows, and the receivers that can send input to the shared application
- Various levels of dialog information (For example, when the user starts using HP SharedX, a Normal dialog level is set giving the user complete status information about the quality of the shared application. When the user sets the dialog level to Experienced, only critical errors are reported.)
- Extensive online help information, including a troubleshooting section for diagnosing and fixing problems when sharing applications.

After the first release of HP SharedX, we gathered information on the product to determine major usability issues. From this analysis three major issues emerged:

- The need to annotate a shared window
- The need for a "shared whiteboard" to assist in collaboration
- The difficulty and security implications of the receiver's granting display access to the sender for shared applications.

The first two usability issues are addressed by the HP SharedX Whiteboard (see "Whiteboard: A New Component of HP SharedX" on page 28). The third issue is addressed by the HP SharedX receiver service (described below), a program that is invoked on the prospective receiver's machine when sharing is attempted.

Receiver Service

When a sharing request is made to the connector via the Share button, the connector software first tries to invoke the receiver service program on the receiver's computer. The receiver service displays a dialog box on the receiver's display (Fig. 3). The dialog box informs the user that the sender wants to share a window and asks for a yes or no response. If the user selects yes, the receiver service grants access to the receiver's display for sharing.

The connector can get one of three responses from the receiver service. The receiver can answer yes, in which case the connector will share the specified window. The receiver can answer no, in which case the connector will display a dialog box informing the sender that access permission was denied. Lastly, the receiver service may not be present, in which case the connector will attempt to share with the receiver's machine.

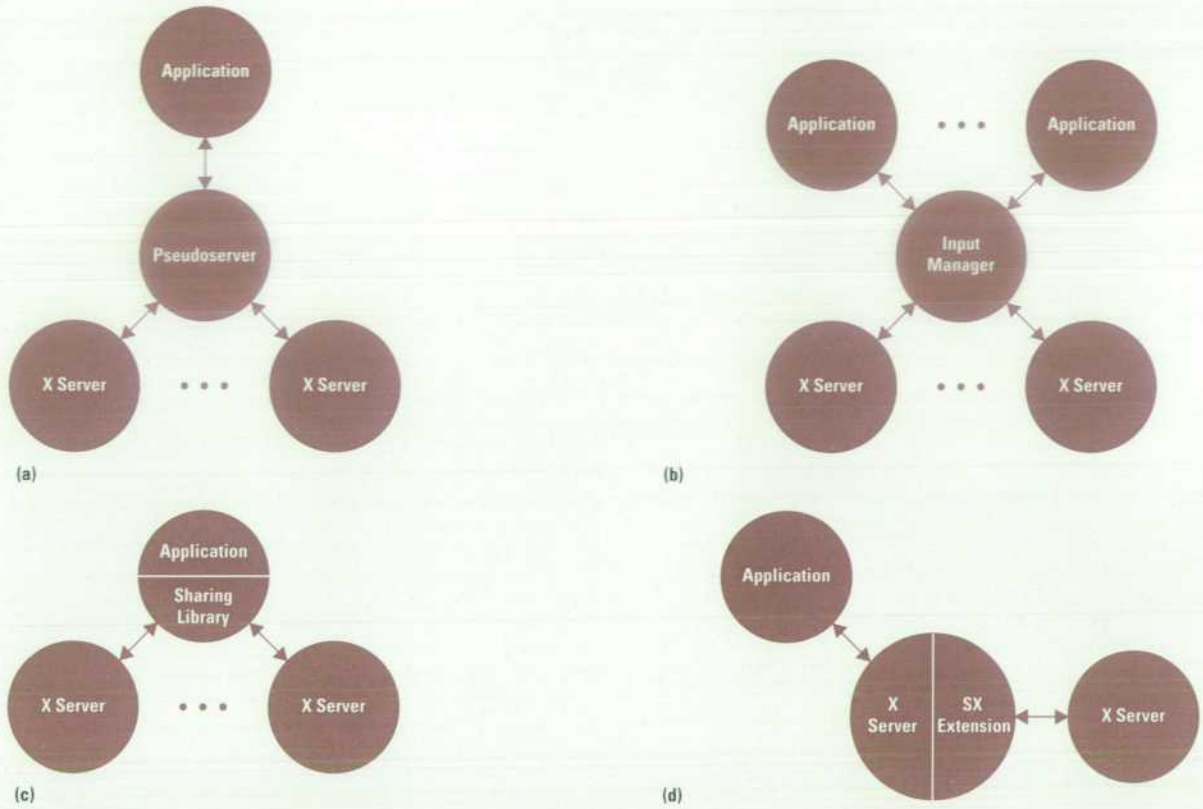


Fig. 5. Different application sharing architectures. (a) Centralized pseudoserver architecture. (b) Replicated architecture. (c) Sharing-library architecture. (d) Integrated architecture.

Once HP SharedX has shared windows to the receiver, it tells the receiver service to remove display access from the sender. This prevents processes other than HP SharedX on the sender from making unauthorized connections to the receiver's display.

HP SharedX Extensions

The HP SharedX extensions are low-level routines responsible for intercepting data streams that flow in and out of the X server during a sharing session. These routines merge input from multiple receivers and ensure that the receivers' displays are updated properly.

Several groups of researchers have attempted application sharing in the X window system environment.[3] These implementations can be classified into four different architectures, each with inherent strengths and weaknesses[4] (see Fig. 5). The main goal of sharing X applications is to ensure that the application and the basic X server don't have to change to work in a sharing environment. This requires some sort of interface mechanism between the application and the X server.

By far the most popular architecture for window sharing in X is the centralized pseudoserver architecture (Fig. 5a). This architecture places a process responsible for output multicasting and input merging between the shared applications and the X server.† This architecture is the most flexible, but suffers from performance problems, even by unshared applications, because it places a process between the application and the X server.

Other application sharing systems have used a replicated architecture for sharing windows (Fig. 5b). The replicated architecture runs a copy of the application for each receiver of the shared application and keeps these copies synchronized by sending identical input to each. Although this architecture optimizes the image for each receiver of the shared application, it has been shown to be difficult to keep the instances of the application synchronized and nearly impossible to add users to an existing sharing session.

The sharing-capable library (Fig. 5c) moves the output multicasting and input merging found in the pseudoserver architecture into the library of X functions (Xlib) linked into the application. By moving the sharing into an existing process and removing the pseudoserver process, performance is enhanced, but applications in such a system cannot be sure of sharability, either because they have not linked with this special library, or because they are running on a machine in the network that does not have the sharing-capable library.

The integrated sharing architecture, which is used in HP SharedX, moves the output multicasting and input merging routines inside the sender's X server (Fig. 5d). This architecture ensures that all applications are sharing-capable, while eliminating the pseudoserver process.

† Output multicasting is the generation of multiple streams of requests from a single stream. Input merging involves coalescing multiple streams of events into a single stream.

(continued on page 29)

Whiteboard: A New Component of HP SharedX

Whiteboard, a general-purpose image viewer and annotation X application, is a new addition to HP SharedX. Whiteboard evolved from users' needs to annotate graphic images in shared applications. Although it is impractical to annotate images in live X applications because of limitations in the X Window System, Whiteboard allows users to share snapshots (portions) of their display and to annotate those snapshots. Not only is Whiteboard useful for adding annotation to computer images, it can also be used to create original images. As an X application, Whiteboard is treated just like any other X application when it is shared in the HP SharedX environment.

Some of the other features that make Whiteboard convenient, powerful, and very functional when shared include:

- Annotation performed on an image can be erased without destroying the original image.
- Any region of the user's screen can be copied and pasted into the Whiteboard drawing area, even if the region contains areas of different X visual types. This ability to copy an arbitrary region containing multiple visual types is a new capability for X applications.
- When shared, Whiteboard knows when input is coming from the sender or receiver. Thus, it can respond differently according to the source of input. This capability is used to ensure that receivers are restricted in what screen regions they can copy to the Whiteboard.

Fig. 1 shows a typical Whiteboard display.

Annotating Images

The main feature that differentiates Whiteboard from other drawing applications is the concept of erasable and unerasable layers. Maintaining two layers allows users to erase image annotations without affecting the image being annotated.

When an image is loaded into Whiteboard, it is in the unerasable layer. Annotation can be performed in the erasable layer and then erased without destroying the original image. This is analogous to drawing an image on a real-world whiteboard in indelible ink, annotating the image in erasable ink, and then erasing the board leaving the original image. This user model is convenient in collaborative situations where changing just the image annotation is desired. For convenience, Whiteboard allows annotation to be hidden temporarily to reveal the unerasable image.

Copying Multivisual Regions

Copying a region from the screen is not as trivial as one might suspect. To an X programmer, using the Xlib function **XCopyArea** to copy a root window region into an **XPixmap** buffer might seem like an obvious solution. However, **XCopyArea** cannot handle cases in which the selected region includes areas of multiple visual types or visual types different from that of Whiteboard; the source and destination visual types must be the same to use **XCopyArea**. In more recent X displays, a screen can simultaneously support multiple visual types differing in depth and types of color maps. Therefore, it is possible to have child windows of differing visual types in the window hierarchy, which is not an uncommon situation.

An algorithm was developed to convert all image data in the selected region to the destination visual type. This algorithm involves scanning the selected region for all windows and parts of windows, coalescing those of the same visual types into subregions, then reading and converting each subregion to the corresponding subregion of the destination pixmap (paste buffer). If any subregion in the selected region matches the visual type of the buffer, a simple **XCopyArea** call is used to transfer the image information. Otherwise, the conversion algorithm uses functions in HP's Image Library to convert each subregion to the destination visual type.

If the X server on which Whiteboard is running supports deeper visual types than the default, Whiteboard keeps the paste buffer in the deepest visual type available, even though Whiteboard is running in the default (shallower) visual type. This technique maintains the buffer contents at the highest possible color fidelity, so that when a scaled paste is done, making the image larger, the resulting image does not contain pronounced dither patterns.

Recognizing the Source of User Input

Whiteboard is the first application that is able to detect the source of user input when the program is shared. Whiteboard uses a feature of the HP SharedX extensions to the X server on the sender to examine events and determine the source of input. Based on whether the sender or receiver generates the input, Whiteboard performs a different operation when the **Copy Region** button is pressed. When the sender presses the **Copy Region** button, Whiteboard allows the user to select a region on the sender's screen to copy. However, for security reasons a receiver should not be able to copy the sender's whole display. Instead, a **Copy Region** operation by the receiver confines the function to the Whiteboard drawing area.

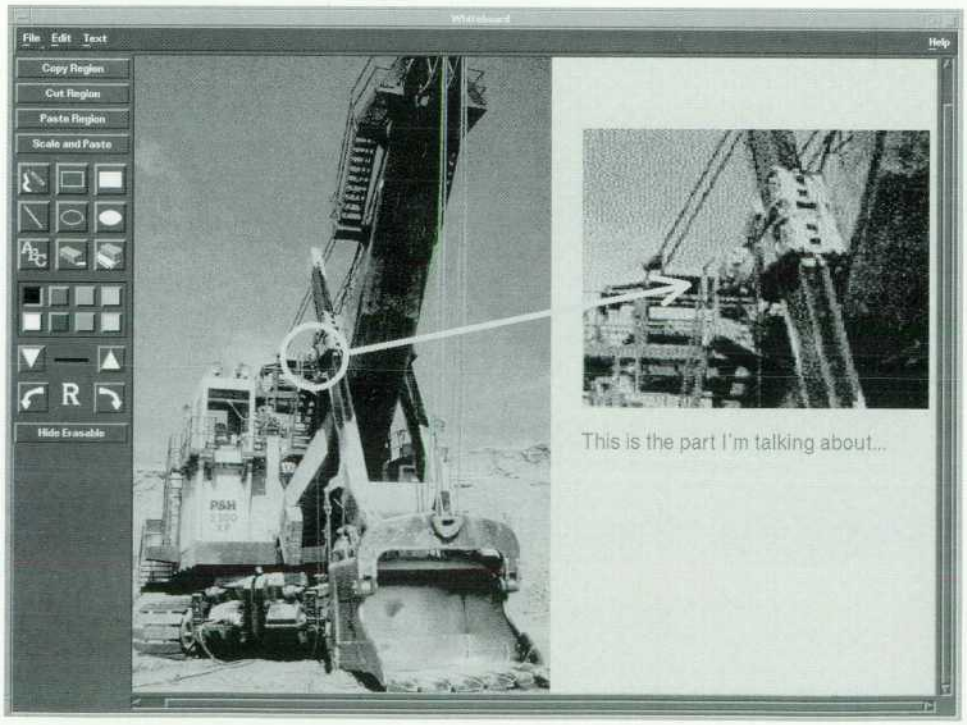


Fig. 1. A typical Whiteboard display. The white arrow with the circle on the end and the text illustrate a typical annotation. The picture on the right, which shows a zoomed-in portion of the main image, is an example of copying and pasting a portion of an image into the Whiteboard.

Ideally, Whiteboard should perform the copy from the display from which input is initiated. Unfortunately, because Whiteboard does not have a direct connection to the receiver's display when the application is shared, no copying operations can be initiated from the receiver's screen. Although the implemented Copy Region function does not present the most ideal solution, it does maintain complete functionality for the sender while providing the receiver with a reasonable substitute for the copy operation.

The advantages of the HP SharedX extension architecture (integrated sharing architecture) include:

- Any X application can be shared at any time
- Unshared applications do not suffer a performance penalty
- Sharing performance is optimized because state information about the X server is directly accessible to the sharing mechanism. (Other implementations require the sharing mechanism to query the X server state over the network, slowing down the sharing process.)

The main disadvantage of the HP SharedX architecture is the need to modify the X server, which requires access to the source code for the X server. Using the sharing-library approach, an application can be linked with a sharing-capable Xlib and can then be run with any standard X server.

Although these architectures for sharing applications differ, the basics of sharing an application are the same for each (with the exception of the replicated architecture). These basics include making instances of windows on receivers' displays, echoing the rendering requests of an application to each receiver, matching resources on the sender machine with those on the receiving machines, and merging input events from the multiple X servers into a single stream and returning the information back to the application. The following sections give a detailed look at how HP SharedX deals with some of these issues in sharing applications.

Limitations of HP SharedX

Applications are displayed on HP workstations in one of two ways. They are either displayed via the X server using X protocol, or displayed directly using direct hardware access (DHA).[5] DHA allows applications to bypass the X server to render graphics on the display. Since HP SharedX operates by retransmitting X protocol to the receiver's display, applications based on X can be shared, while DHA programs cannot be shared directly. The static images from DHA applications can be shared by copying their windows into the Whiteboard application described on page 28.

Programs that render through the DHA mechanism include those that use the Starbase graphics package or PEX† programming interface. However, the programming interface does not make the final determination of the rendering path. For example, Starbase programs can be run with a graphics driver that emits X protocol instead of using DHA. While there is usually some performance penalty for using these drivers, it does allow the application to be shared.

Even if an application is based on X, it may not share perfectly. For example, the audio application of HP MPower uses X protocol to display its control panel but interacts with the audio server to generate sound. Even though the control panel shares properly, when the receiver presses the play button, the audio is played only on the sender's audio server (see Fig. 6).

† PEX are 3D enhancements to the X protocol.

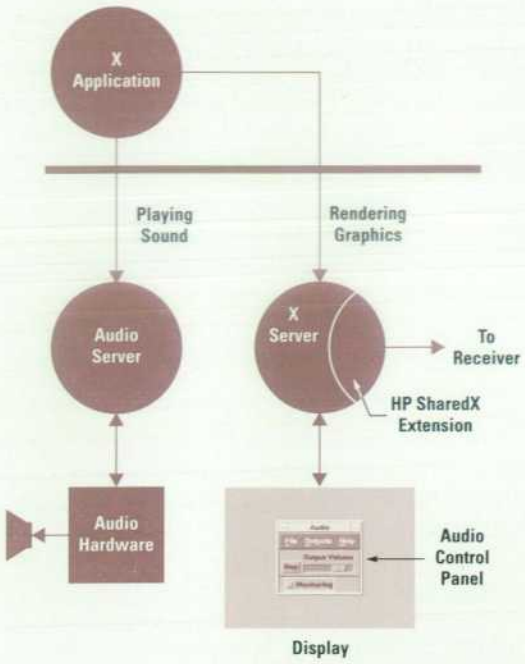


Fig. 6. An HP SharedX configuration in which output from a shared application is not echoed on the receiver. In this case the output is going to the audio server which knows nothing about the receiver.

Another example of applications that do not share perfectly are those applications that use X extensions. The X extension is a mechanism for extending the capability of the X Window System. While the core X protocol is standard among all X servers, X extensions vary among X server implementations. HP SharedX only retransmits core X protocol, so applications that use X extensions will have limitations when shared. For example, if an application uses input devices other than the keyboard and mouse via the X input extension,†† the receiver will see a copy of the window but will not be able to interact with the application with these additional input devices.

There are parts of the core X protocol that HP SharedX cannot handle properly. One example is the X mechanism for cut and paste. This mechanism uses a standard interprocess messaging system for transferring data from one X program to another. However, a receiver cannot cut and paste between a shared application and one on the local machine. The two applications cannot communicate since they are not connected directly to the same X server.

Establishing and Maintaining Connections

When a share request is made to a receiver, HP SharedX must first establish an X connection to the receiver's X server. It is over this connection that HP SharedX will manage the shared windows, retransmit rendering requests, and get receiver input. The display connection is made using the X library (Xlib).

Xlib is the client-side library of functions used by all X applications. These functions create the protocol requests that

†† The X input extension allows applications to receive input from input devices such as graphics tablets, knob boxes, button boxes, and so on.

become the X protocol packets sent over the display connection to the X server to request creation of windows, drawing to those windows, and so on. Although the HP SharedX extension is in the sender's X server, it communicates with the receiver's X server using Xlib. This makes the sender's X server appear to the receiver's X server the same as any other X client.

The HP SharedX extension, however, has some requirements that are different from other X clients, so we made some changes to Xlib to accommodate these requirements. The first change we made to Xlib was to create a mechanism to recover from broken display connections. In X, programs that lose their connection to the X server normally print an error message and exit. For most X programs, their one connection to the X server is their lifeblood, and if it is broken, they may as well terminate. The sender's X server still has plenty to do if it loses a sharing connection, and thus it should definitely not terminate. Xlib calls a user-specified routine, XIOErrorHandler, when a display connection is broken. In HP SharedX's version of XIOErrorHandler, the routine cleans up data structures related to the broken display connection, sends a notification to the HP SharedX user interface, and jumps back to a location inside the X server to continue processing.

A similar problem occurs when the remote X server is not responding. This can happen for a variety of reasons such as that another program may have taken exclusive access to the X server, the network between the two computers may be very busy, or the receiver's X server may be busy handling requests from other clients. In any case, X programs wait indefinitely for requests to be serviced before proceeding, which is not the desired behavior for the sender's X server. Therefore, HP SharedX has added a timeout handler to Xlib that allows the sender's X server to recover from a nonresponsive receiver. After 30 seconds of waiting on a receiver's X server, HP SharedX closes the display connection and notifies the sender that the receiver's system is not responding.

The final change we made to Xlib handles failures to establish an X (display) connection. Normally when a display connection fails, the program is not given a reason for the failure. A mechanism was added to extract the reason for display connection failure from Xlib and to return that information to the user interface. This information allows the user to diagnose problems more easily when attempting to share windows.

Managing Shared Windows

When users share an application, they usually have a clear idea of what constitutes the application and what set of windows should be shared. Most applications consist of a single program with a single connection to the X server. For these programs, it is easy for HP SharedX to share the right windows. However, an application can consist of more than one program (e.g., HP SoftBench), or a single program can display several windows that should not be grouped together (e.g., the HP VUE file manager can display multiple windows, each looking at a different directory). A user of the HP VUE file manager usually wants just one window to be shared, while a user of HP SoftBench may expect the debugger to be shared along with the static analyzer.

From the point of view of the X server, windows partake of two relationships. One is a parent-child relationship, which defines a window tree with all windows as descendants of the root window. The other relationship is that in which each window is owned by a display connection. Each window is uniquely identified by its window identifier, a 32-bit number in which seven of the bits are the same for windows owned by the same client. These seven bits are called the resource base.

Applications typically consist of one or more top-level windows in which applications display information. Because windows are relatively inexpensive to create, they are used extensively inside an application's top-level window for everything from drawing areas to buttons and menus.

When the user selects an application to share, HP SharedX must select a group of windows that best represent the application to send to the receiver's display. HP SharedX makes two assumptions: all top-level windows belonging to a single client connection (i.e., having the same resource base) belong to a single application, and all child windows of a shared top-level window should be shared with that top-level window.

Using these assumptions, if the sender shares an HP VUE file manager window, HP SharedX will send to a receiver all file manager windows. Users have the option of selecting Share Alone from a pull-down menu, but they could easily forget which applications to do this for. However, HP SharedX can be configured to respond differently to the main Share button for different applications. For many applications, HP SharedX is shipped preconfigured to share only the application's selected top-level window and its children.

The other problem mentioned above, that of multiple programs that should be shared together, has not been addressed directly. While application developers could use the HP SharedX command line interface to tie their applications together, HP SharedX does not provide a simple mechanism for grouping windows from different programs. Typically, for these applications the user must manually share all the desired windows.

Input Management

HP SharedX allows both the sender and the receivers of a shared application to interact with an application. In X, application input consists of events from the user's input devices and queries that the application makes about these devices. For example, the application can query the position of the mouse cursor, which, if the application is shared, has as many values as there are viewers of the application.

Most application-sharing systems implement one of two input merging policies: floor passing or free-for-all. The floor-passing scheme allows one user at a time to input to the application, with the choice of the user who has the floor manually controlled by a moderator. The floor-passing scheme ensures that input is not mixed from multiple users, but it requires explicit user action to change the input floor. The free-for-all policy allows anyone to input without explicit action, but it is prone to intermixed input from multiple users.

The HP SharedX input model, called *dynamic floor passing*, is a hybrid of these two methods. In this scheme, only one user at a time can give input to the shared application. Thus,

user input does not become intermixed. However, input may change among users dynamically without explicit action, but only after a specific period of inactivity by the current user giving input.

In addition to answering the problem of intermixed input, dynamic floor passing solves the problem of input queries. This method always returns the state of the user's input devices currently or most recently interacting with the application.

The final challenge of handling input from multiple users is translating the actual input event that occurred on a receiver into an event that could have occurred on the sender. Translating a window identifier is trivial because HP SharedX maintains a mapping of which receiver windows correspond to which sender windows. Translating the keycodes in the event message is more difficult.

The keycode specified in an event is an index into a mapping table of logical key symbols that can vary from one X server to another. For example, one server can map keycode 52 to the letter "a", while another server can map the same keycode to the letter "z". The keycode mapping table is loaded into the shared application when the display connection is made, but the mapping can be changed dynamically.

When an event with a keycode is received by HP SharedX from a receiver, it is translated into a key symbol based on the current mapping for the receiver's X server. HP SharedX searches the sender's keyboard mapping to see if any keycode matches that same key symbol. If it does, the matching keycode is returned in an input event to the application. If no corresponding keycode on the sender's machine exists, the key event is discarded since no logical keycode can be sent to the application.

The flow diagram in Fig. 7 shows the operation of the input merging routine, including dynamic floor passing control and event-code translation.

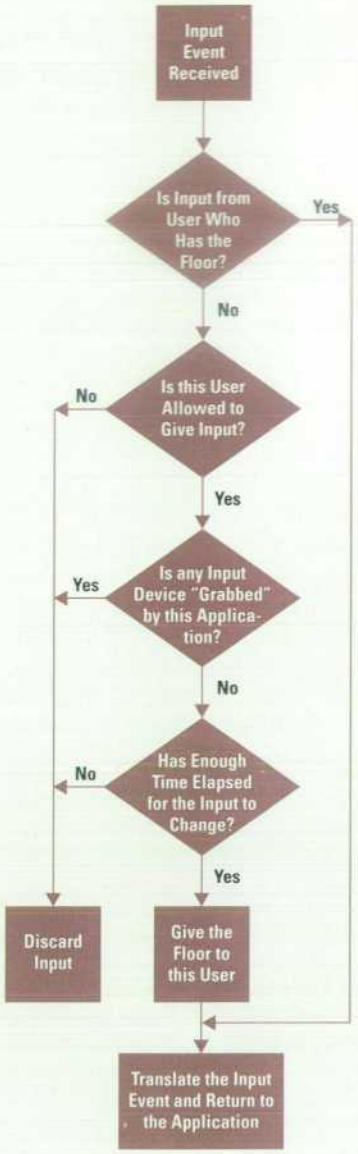


Fig. 7. Flowchart for the HP SharedX input merging routine.

Allocating Display Resources and Rendering

HP SharedX uses a "lazy" allocation scheme for display resources such as colors, pixmaps, graphics contexts, and fonts. To minimize initial sharing time and the impact on the receiving machine, display resources are allocated only when needed for a rendering request. For example, a shared application may allocate a large pixmap that is only displayed for some error condition. If that error condition never occurs during the sharing session, it would be a waste of time and resources to make an instance of the pixmap on the receiver's machine.

Once allocated, the display resource mapping is maintained so that future requests for that same display resource are satisfied without contacting the receiving X server. For example, if an application requests a line be drawn to a shared window, HP SharedX performs the following procedure:

```
draw the line to the local display;
for each remote instance of the window
begin
    if a remote instance of the graphic context has already
    been allocated,
      use that instance,
    else
      allocate a new remote graphic context and map it to
      the local graphic context;
    draw a line to the window instance with the graphic
    context instance;
end;
```

When the allocation of a display resource fails, HP SharedX uses several means to recover. In the case of pixmaps, HP SharedX continues operation of the application without the use of that image and notifies the user of this situation. In other cases, HP SharedX will substitute other display resources for the ones it cannot allocate. The next two sections provide detailed examples of two display resources HP SharedX will gracefully degrade: colors and fonts.

Managing Colors

The management of display resources such as graphic contexts and windows in HP SharedX is fairly straightforward compared to the management of colors. The goal of displaying identical images on all shared windows requires mapping colors from the sender's display to the receiver's display and degrading the colors gracefully if the exact colors are unavailable.

The term pixel refers to the number that represents a color value. The pixel can be one of two types: read-only, which represents a color that does not change and can therefore be accessed by multiple programs simultaneously, or read/write, which can be changed and is exclusively used by a single application.

Pixels are converted to colors either by using the pixel as an index into a color map (a look-up table) or by treating the pixel as a representation of the color itself. The interpretation of the pixel is based on the visual type,† of which there are six in the X window system. Three of these allow X programs to allocate read/write cells (grayscale, pseudo color, and direct color), while the other three provide a fixed set of colors (static gray, static color, and true color).

To divide the six visual types another way, four of them (static gray, grayscale, static color, and pseudo color) use a single color map to map a pixel into a gray value or an RGB triplet (Figs. 8a and 8b). These displays typically use between one and eight planes, so their color maps typically store between 2 and 256 color values. Another visual type, direct color, uses three color maps, one each for red, green, and blue. The pixel is decomposed into three indexes for look-up in the three color maps (see Fig. 8c). The last, true color, decomposes the pixel directly into red, green, and blue values (see Fig. 8d). True color and direct color typically use either 12 or 24 planes, yielding 4,096 or 16,777,216 colors.

† Color map capabilities (see glossary on page 26).

Our color mapping solution addresses two questions. Given the sender's and receiver's visual types, and the sizes of their color maps (number of different colors available):

- How are the available colors used to best advantage?
- How is color mapping maintained?

Using the Best Colors Available. The key to color matching is always to have a color that is close enough if the exact color is unavailable. If the receiver's color display system is static, supporting only read-only colors, the solution is simple: map each sender color into the closest receiver color. For receivers with dynamic color systems that support read/write colors, a color ramp is created. The color ramp contains a set of colors evenly distributed throughout the RGB color space, a three-dimensional space defined by axes of red, green, and blue values (see Fig. 9). The benefit of the ramp is that any color matches a ramp color within some maximum color difference in the color space. However, the maximum color difference may still be larger than is desirable, so the ramp is a fallback if an exact color match cannot be allocated.

A color ramp array and the values for the receiver's color map are created when the display connection is made to the receiver for sharing. Fig. 10 shows an example of a color

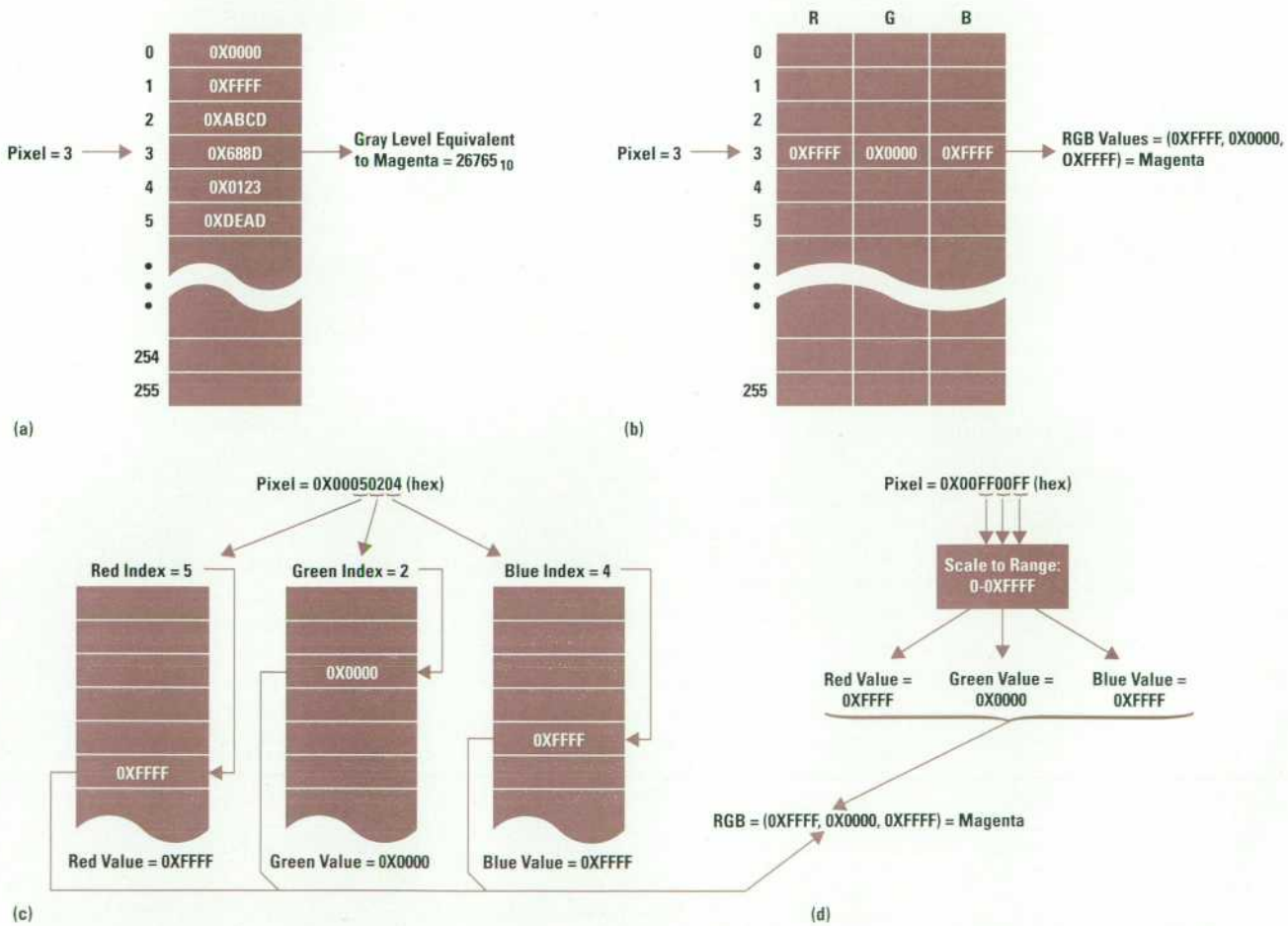


Fig. 8. Four representations for the color magenta. (a) The read-only gray level (static gray or grayscale) equivalents for magenta. (b) Read-only static color or pseudo color representations. (c) Direct color representation of magenta. The pixel is split into indexes into the color map. (d) True color representation. The pixel value is split into parts, and each part is scaled to create the values for the different shades of red, green, and blue for magenta.

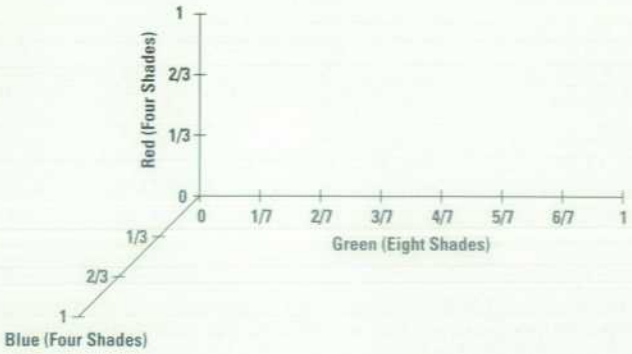


Fig. 9. A three-dimensional representation of the color ramp for an 8-bit color display with four shades of blue, four shades of red, and eight shades of green.

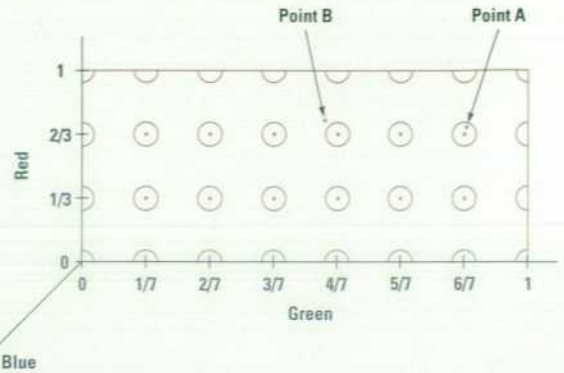


Fig. 11. A two-dimensional representation of the color space used by the color-matching algorithm.

ramp array that contains pointers to the color map on the receiver. The values in the color map are determined from the three-dimensional color ramp mentioned above. As each entry is allocated in the receiver's color map, the index to that entry is sent back to the sender and placed into the ramp array table, creating a table of indexes into the receiver's color map.

The size of the color ramp depends on the size and type of the destination visual type. For grayscale visual types, a grayscale ramp of up to 16 values is allocated. For direct color visuals, up to 16 levels of red, green, and blue for a total of 4,096 distinct color values are allocated. In both of these visual types, 16 levels provide enough resolution to ensure a good color match without using excessive color map entries. The ramp used for eight-plane pseudo color consists of all combinations of four evenly spaced values of red, eight values of green, and four values of blue, for a total of 4 by 8 by 4 = 128 colors. The ramp takes up half of the available colors in an eight-plane color map.

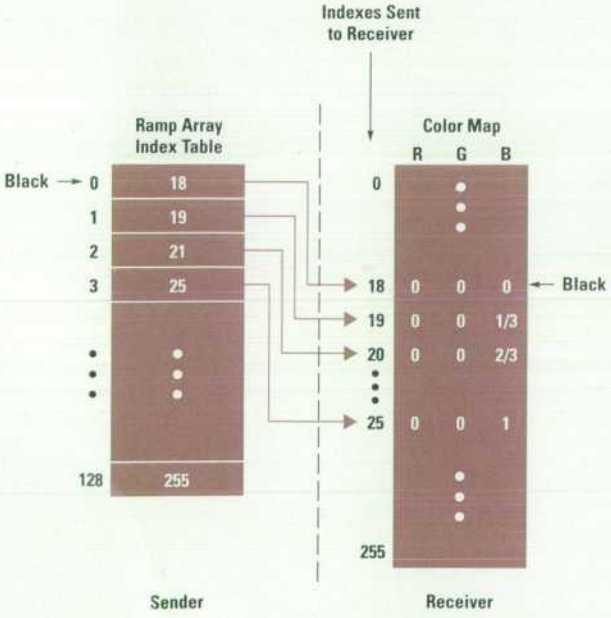


Fig. 10. The color ramp array containing indexes into the receiver's color map. Both of these items are created when the connection between sender and receiver is first established.

Eight-plane pseudo color visual types are the most common and most difficult type to deal with since not enough colors exist to cover the color space adequately with a ramp. To compensate for large color errors, HP SharedX judiciously uses the remaining color cells available in addition to the ramp. For any given color, the following color-matching algorithm is used to map the sender's colors onto colors on the receiver:

1. If the X application on the sender allocated the color as read/write, then attempt to allocate a matching read/write color on the receiver. A read/write color is needed since the application may change the color of the allocated pixel at some later time and, to maintain color accuracy, the receiver's matching color must be changed as well.

2. If the application allocated a read-only color or if the above allocation failed, find the closest color from the ramp or a previously allocated read-only color. If that color is close enough to the desired color, use it as the match.

3. If the closest read-only color is not close enough, try to allocate a new read-only color. If this succeeds, add the color to the list of allocated read-only colors and return the new pixel as the match.

4. If the read-only color could not be allocated, use the closest pixel from step 2 since it is the best available match. Notify the user that the colors do not match exactly.

Fig. 11 shows a two-dimensional representation of the color space given in Fig. 9. Each circle encompasses colors that are close enough to a particular color ramp value (represented by the point in the middle of each circle). Each of the points represents a read-only color that has already been allocated on the receiver's X server.

The color-matching algorithm first checks to see if a color is close enough to an already allocated color. Point A in Fig. 11 is close enough so another cell is not allocated (step 2 in algorithm). Thus, for point A, 2/3 red, 6/7 green, and some value for blue is used.

Point B in Fig. 11 is not close enough to any allocated color, so a color is allocated and point B is added to the list of available colors (step 3 of the algorithm). If B cannot be allocated in the color map, the closest color is selected that has already been allocated, although technically it is not close enough (step 4 of algorithm).

The difference between the desired color and the candidate color is measured as the sum of squares:

$$\begin{aligned}\text{diff} = {} & (\text{desired\_red} - \text{actual\_red})^2 + \\ & (\text{desired\_green} - \text{actual\_green})^2 + \\ & (\text{desired\_blue} - \text{actual\_blue})^2\end{aligned}$$

The color with the minimum difference is the closest color. A close enough threshold was determined empirically to optimize the accuracy of color images while minimizing the number of receiver color cells used. Since images tend to have color themes (i.e., the colors are clustered in a few regions of the color space), a fairly high degree of color accuracy can be obtained without excessive demand for color cells.

Private Color Maps. An X display has a default, shared color map that most applications use. If an application needs more colors than are available in the default color map, it can create and use a private color map over which it has complete control. When an application using a private color map has its window focused, that application's color map is installed (copied into the display hardware) by the X window manager. On displays that support only one color map in hardware (most of the low-end displays), everything on the entire screen is displayed using the installed color map. When a private color map is installed, all applications using the default color map take on random colors. As soon as an application using the default color map gains the focus, the default color map is reinstalled, and the application with the private color map will have random colors. As the current color map switches back and forth from default to private, the user sees color flashing. The user typically prefers to display shared windows with the default color map to avoid this irritation.

When a window is shared to receivers with display types that have read/write color maps, a color ramp is created on the receiver's X display. The ramp is created in the default color map to reduce the probability of color flashing. If the desired ramp cannot fit in the default color map, it is placed in a private color map.

The system user interface is usually the first process to allocate pixels, and the pixels with the smallest indexes are allocated first. When a private color map is used, HP SharedX copies some of the lower pixel values from the default color map into the private color map. This way, when X switches between the two color maps, the colors used by the system user interface do not flash. The number of pixels copied is optimized to reduce color flashing while leaving color cells for later allocation.

Keeping Track of Pixel Mapping. To minimize the number of times the color matching algorithm is executed, a mapping of previously matched pixels is maintained. Different mapping methods are used depending on the range of possible sender color values. For small ranges (up to 256 different colors) a simple array is used, in which:

```
receiver_pixel = map [sender_pixel]
```

When the sender is a 24-plane display, this approach would require 48 megabytes of memory, so a more memory-efficient mapping scheme is needed. Since the goal is to produce close colors for the receiver rather than exact color matches, resolution of the sender color is reduced before applying the color mapping. Red, green, and blue values of the color are reduced to four bits rather than eight bits for each plane, which results in 16 (2^4) levels of each. Thus, the total number of possible color values is 4096 (16^3), a reasonable size for a mapping array. If color() is a function that returns the RGB values of a pixel, and crunch() is a function that converts an RGB triplet to a number in the range 0...4095 (four bits per pixel), then the mapping is as follows:

```
receiver_pixel = map [crunch (color (sender_pixel))]
```

The result of this mapping is that all colors close to a mapped color are mapped with that color. This method, called color-zone mapping, gives fast performance and adequate color matching, while keeping memory use fairly small.

Matching Fonts

The fonts in which applications display text can be specified by the user in the X Window System. From the set of fonts supported by the X server, the user selects a font that is aesthetically pleasing. Since there are no standard fonts in X, each X server can support a different set of fonts.

To maintain consistency between the sender and receiver of a shared application, it is important that text be displayed in a similar font on the receiver's display. HP SharedX employs a font matching algorithm to ensure a close font match.

Fonts in X have both a name and characteristics. A font can be loaded by specifying either its name or its characteristics using the X Logical Font Description (XLFD) format. HP SharedX first tries to match the font by name since it is unlikely that fonts with the same name differ. If the font with the same name is unavailable on the receiver's X server, fonts are matched by their characteristics.

The XLFD description defines 13 characteristics of a font such as its size, character set, weight, slant, and style. When matching fonts for the purpose of sharing an application, some of these characteristics are more important than others in choosing the font with the closest characteristics.

HP SharedX obtains a list of all XLFD type fonts from the receiver's X server at display connection time. When a font match is needed, the source font's characteristics are compared to those available on the receiver's X server. A weighted sum of differences of the characteristics is calculated and the font with the minimum difference is considered the closest match.

The weighting of characteristics was determined through a combination of intuition and observation. With the exception of character set, discussed below, it seems clear that size is the most important criterion, with width taking precedence over height. This is partly because many X applications write text in two different ways. One way is to pass a whole string of characters to the server and let it determine the location of each character based on its knowledge of the font. The other way is for the program to control the spacing and send one character at a time to the X server. The program must then know about the character widths for the font in use. In this method, the program has no knowledge that the receiver may be using a different font with different spacings, and characters may overlap or be widely spaced for their size. If the first method is used, the receiver's X server will correctly space the characters for the font in use, but the positions of characters within the window will be

incorrect. If both methods are used, as they are in terminal emulators, the receiver is faced with a messy mixture of incorrect spacings and characters in incorrect positions.

Another important factor with respect to spacing is whether a font is proportional or monospaced. Proportional fonts use different widths for different characters, while a monospaced font uses the same space for all characters. In most cases, it is better to match a monospaced font to another monospaced font, even if they are of slightly different size, than to match it with a proportional font of the same size. Likewise, matching a proportional font with another proportional font tends to give the best appearance, and if the typefaces are similar, the chance is high that specific character widths will be similar.

The issue of character sets is an easy one for an English speaking person to ignore. Almost all fonts have identical characters in the range of characters most often used (the ASCII characters) numbered 32 to 127. The differences lie mainly in the upper range, the characters numbered 128 to 255 and beyond. This is where characters with special accents, used in most European languages, are found. If accented characters are used, the difference in character sets is very important.

This issue becomes more important when character sets for pictograms or entirely different alphabets are used. If the character set does not match, the receiver is given meaningless garbage. But what constitutes a match? Does the name of the character set have to match exactly, or are there more or less equivalent character sets that have different names? These issues have not been addressed in the current font matching algorithm. For HP SharedX to work acceptably with these types of characters, it is best to have the same font on sender and receiver machines.

Performance

The performance of HP SharedX depends on the characteristics of the network connection, the performance of the workstations involved, the application being shared, and the operations performed with the application. There are three networking factors that affect performance: network speed or bandwidth, line delays, and network load. Increased load is roughly equivalent to decreased bandwidth. Performance increases directly with network bandwidth, up to a limit of about 500 kbits/s for an unloaded network. Beyond that, other factors limit performance. Line delays are of greater importance when sharing over a wide area network (WAN). Screen update times increase proportionally to line delays.

Given a network with reasonable bandwidth, HP SharedX performance will be greatly affected by the performance of the computers involved in the sharing, especially the sending machine. When users have a choice of who sends and who receives, the person with the faster machine should be the sender.

Applications vary widely in how they make use of X. For example, an application receives an expose event from the X server, indicating that some portion of a window has just become visible and therefore needs redrawing. Some applications redraw the entire window, while others are more efficient and only repaint the portion that is exposed. A word processing application may update the entire window every time the user types a letter. While this scheme works acceptably when not sharing, this application is nearly unusable when shared.

If application windows can be resized, the sender can improve performance by making windows smaller. Also, users often come to recognize slow operations. If resizing can be done before starting the sharing session, there will be less demand on the network. If the application is still too slow when shared, the sender can take snapshots of it with the Whiteboard and share the Whiteboard.

Lazy allocation of resources in HP SharedX affects performance in some interesting ways. For example, the Whiteboard uses two large pixmaps for the drawing area, one for the erasable layer and one for the unerasable layer. The erasable pixmap is allocated immediately when the Whiteboard is shared. The other, however, is not used until the user does an erase operation. The user may be typing some text into the drawing and then hit the backspace key. At that point, the Whiteboard and all other X programs on the sender's display "freeze" while the unerasable pixmap is created on the receiver's X server.

When more than one receiver is involved in a sharing session, the method of connection becomes important. If there is one receiver, and the parties involved want to add a second, either the sender or the receiver can share to the new person. The first receiver sharing to the second receiver is called daisychaining. The sender sharing to more than one receiver is called fanning out. With the right combination of daisychaining and fanning out, an application can be shared to a large number of people.

Determining an optimal configuration for one-to-many sharing is more complex when the computers vary in performance or when the network links between the parties vary. Some general rules for optimizing performance are:

- The fastest machines should be daisychaining, and slow machines should all be leaves in the sharing tree.
- When some of the receivers are connected by a LAN, while others can only be reached over a WAN, the number of WAN connections should be minimized.

For example, if a sender in Colorado is sharing a window to five receivers in New York and three receivers in California, it is most efficient for the sender to share the window to the fastest receiver in New York and the fastest receiver in California and have those receivers share to others on the same local network.

Conclusions

The implementation of HP SharedX presented several problems. First and foremost, since this type of technology is new, no foundation of past experience was available to build upon, especially when it came to designing the HP SharedX user interface. The user interface design challenges were solved by applying human factors design techniques, including user task analysis and human factors testing. The HP SharedX extension presented a totally different challenge, including handling input from multiple sources in a sane manner and applying X protocol customized for a machine of one type and mapping it to machines of very different types. The HP SharedX extension challenges were met by

understanding the nature of the X window system and gracefully degrading the display image when receiver display resources do not match those of the sender's display. Although there are no perfect answers to any of these challenges, the value of HP SharedX far outweighs its limitations.

Acknowledgments

Many individuals have contributed to HP SharedX, from its conception at HP Labs to its becoming a product. Steve Lowder and Phil Gust played a major role in initiating the HP venture into collaborative tools and helped with the initial HP SharedX prototype. Thanks also to Nancy Kedzerski and Dan Flickinger; HP SharedX would have never become a product without their support. Special thanks to Joe Gersch, whose vision carried HP SharedX from an HP Labs prototype to a commercial product. Thanks to Randy Branson in product marketing for his valuable input and support. Thanks also to the design and implementation team of John Byrnes, Fred Sprague, Rich Wildman, Susan Frontczak, Perry Wells, Jeff Wood, Ken Burgess, Steve Wolf, and Mary Jones. Thanks to Bob Benusa for information on HP SharedX performance. Finally, thanks to Jan Ryles for her insight into users and tasks.

References

1. F. E. Hall and J. B. Byers, "X: A Window System Standard for Distributed Computing Environments," *Hewlett-Packard Journal*, Vol. 39, no. 5, October 1988, pp. 46-50.
2. K. H. Bronstein, D.J. Sweetser, and W.R. Yoder, "System Design for Compatibility of a High-Performance Graphics Library and The X Window System," *Hewlett-Packard Journal*, Vol. 40, no. 6, December 1989, pp. 6-12.
3. J. C. Lauwers, "Collaboration Transparency in Desktop Teleconferencing Environments," *Computer Systems Laboratory Technical Report CSL-TR-90-435*, Stanford University, July 1990.
4. D. Garfinkel and R.J. Branson, " A Comparison of Application Sharing Architectures in the X Environment," *Proceedings of Xhibition '91*, June 1991.
5. J. R. Boyton, et al, "Sharing Access to Display Resources in the Starbase/X11 Merge System," *Hewlett-Packard Journal*, Vol. 40, no. 6, December 1989, pp. 22-23.

Imaging Services in a Multimedia Environment

Image manipulation tools, compression and decompression functions, picture quality adjustment techniques, and support for industry standards are some of the features included in the HP Image Library.

by Andrew Munro and Ahmad H. Shekarabi

On UNIX*-system-based computers words are the traditional means of communication whether the user is creating a business report or presentation, writing a functional specification, or sending electronic mail. While the topic might benefit from some visual content, the user usually finds it easier to just use words.

Unfortunately, words may not be enough. The user could be describing CAD graphics that remote colleagues need to see. Perhaps, the user is an insurance adjuster who is sending a report to the home office that describes photographs of damaged property. Another user may be describing the appearance of a computer screen that a customer support engineer needs to see.

Without imaging capabilities on the computer, the user may resort to noncomputer means, such as the postal service, to send images.

This article describes the major parts of the HP imaging solution, how it meets the characteristics required of an imaging system, and its application to HP MPower.

Image Files

Image files contain computer graphics and digital records of physical objects, such as photographs, pages from books, and faxes. The images of these objects are represented as collections of bits called pixels.

Image data comes from several sources including screen captures, video frames, and external devices such as scanners, video recorders, or fax machines. Once the image data is collected in a file it can be displayed, modified and saved, or printed.

The data in image files is stored in different formats. These image formats are more commonly called image types. The most popular image file format is called tagged image file format, or TIFF. The following are some of types of images stored in TIFF files. These image types are listed in order of image quality—from a simple monochrome to the highest-quality color:

- Bitonal. Bitonal images contain pixels with two levels—black and white—but they can be displayed using any two colors. Each pixel takes one bit. Bitonal images are frequently referred to as monochrome images. They often contain text, such as a page of a fax.
- Grayscale. Grayscale images contain pixels that identify levels of gray, from black to white. Grayscale images are sometimes also referred to as monochrome images. They are often used for images of scanned photographs.
- Palette. Palette images contain pixels that index into a color map containing the red, green, and blue values for the pixel. The color map has three contiguous sections: red, green, and blue, each with 256 16-bit entries. The color of a pixel is determined by using the pixel value as an index into each of the red, green, and blue sections to obtain the desired color. This is the lowest-quality color image and is also called a pseudo color image, a type commonly used for computer-generated images.
- RGB. An RGB image contains pixels with three samples: red, green, and blue. Each sample identifies a level of that primary color. This is a good-quality color image type that is typically used for photographs. RGB format provides better picture quality than the palette format because RGB provides 2^{24} color variations for images, whereas the palette provides 2^{8} color variations for images.
- YCbCr. A YCbCr image contains pixels with three samples in the order Y, Cb, Cr. (YCbCr images are often incorrectly termed YUV images.) The Y sample is called the luminance sample; it represents the gray level of each pixel. Cb and Cr are called the chrominance samples. Together with Y, they represent the color of each pixel. An RGB image can have the same high quality as a YCbCr image, but a YCbCr image often requires less disk space.

These image types are supported by the image libraries described in this article.

Using Image Files

In trying to provide a software alternative that makes using images as easy as using text, an application developer is usually faced with the following issues from potential customers.

- I don't have enough disk space to store images.
- If I compress my images, won't they become slow to display?
- How can I clearly display photographs when the screen's resolution is only 5% of the photograph's resolution?
- Aren't there too many computer formats of image files to deal with?

Because of these issues, Hewlett-Packard took a comprehensive look at the imaging requirements of end users and application programmers. From this investigation, the image library project team determined that an imaging solution with the following characteristics would be needed to fulfill the needs of both application developers and end users:

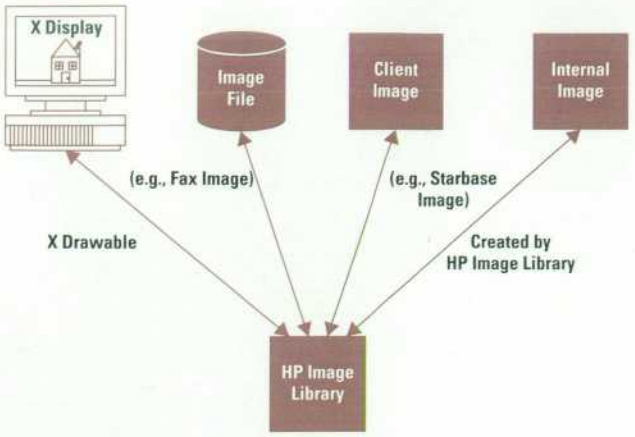


Fig. 1. The four types of images the HP Image Library can read, write, and display.

- Support for industry-standard image file formats
- Minimum disk space requirements for storing images
- Fast, high-quality display of compressed and uncompressed images
- Basic image manipulation, such as rotating and cropping
- Extensibility to allow programmers to add custom image functions and custom image file formats.

The solution we created to help fulfill these needs has three parts: libraries of image functions built on the X Window System, end-user tools built on these libraries, and an image developer's kit with an extensible application interface.

HP Image Libraries

To make imaging functionality a standard capability on HP 9000 Series 700 workstations and Series 800 multiuser systems, the image team decided to create two libraries and build them into the standard run-time environment. These two libraries, the *image library* and the *extensible file support library*, contain high-level functions that C programs can use to access and manipulate images. These two libraries are collectively called the HP Image Library. With the HP Image Library functions, applications can read and write images that exist in four forms:

- A file image. This is an image in a single or multipage file (e.g., a file containing the image of a fax cover page plus one or more other fax pages).
- A client image. This is an image in memory that is created and managed by a client application separately from the HP Image Library functions.
- An internal image. This is an image in memory that is created and managed by the HP Image Library functions.
- X Drawable. An X window image such as an HP terminal window or an X pixmap such as an icon symbol.

Fig. 1 summarizes the relationship between these four forms of images and the HP Image Library.

While the image library functions support TIFF image files, the extensible file support library addresses support of non-TIFF image files. The extensible file support functions operate on image files in an object-oriented approach, independently of the file type. Therefore, if the programmer defines a new type of image file, the existing client programs continue to work on the image file without modification.

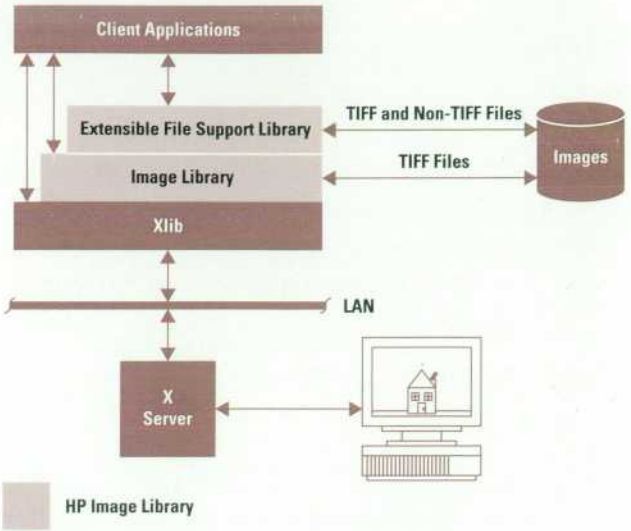


Fig. 2. The image library and extensible file support library in the client/server architecture.

Although the extensible file support library also supports TIFF, that support is not as extensive as the image library support of TIFF. The extensible file support of TIFF files enables applications to treat TIFF files as merely another image file.

The HP Image Library runs on the HP-UX* operating system and uses the X Window System to display images (see Fig. 2). Because the image functionality is layered on standard X, the user gains all of the client/server advantages of X.

Using Pipes to Access Images

To simplify the programming task, the image team designed the image library and the extensible file support functions around the concept of an image pipe. An image pipe is a series of calls to functions, beginning with a producer (a function that reads an image from a source such as an image file) and ending with a consumer (a function that writes an image to a destination such as a display). Fig. 3 shows the steps in a sample image pipe.

While only one producer and usually only one consumer function are allowed per pipe, the pipe can also contain functions that manipulate the image. These filter functions perform operations such as decompressing, scaling, rotating, or converting the image.

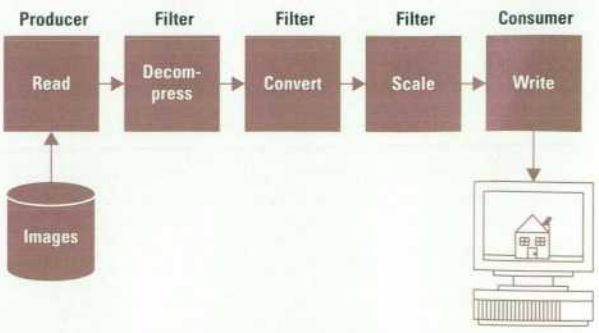


Fig. 3. The steps in an image pipeline.

A producer can access any form of image (Xwd, file image, client image, or internal image). Filters operate on the image data supplied by the producer. The consumer receives the filtered image and writes it to the display, a file, or an internal or client image.

When the pipe is executed, it processes the image in horizontal slices, which are called strips. Processing strips requires smaller buffers between filters and less memory than processing an entire image. Because processing strips can be located in cache memory, image processing performance benefits. For complex image processing, images begin to appear quickly because there is no need to wait for the entire image to be processed before displaying it.

Industry Standard File Types

Since there are several sources for images (e.g., scanners, PCs, fax machines, LAN, and so on) multiple types of image file formats have emerged.

To enable applications to handle these different file types, the HP Image Library provides reading and writing support for the image file types listed in Table I.

The following are definitions for file types not defined earlier.

- LZW. Lempel-Ziv and Welch compression format (described below)
- Xwd. X window (an image created by storing the contents of an X window in a file)
- GIF. A common format for palette images
- JPEG. Joint Photographic Expert Group compression, a lossy compression format
- JFIF. JPEG compression that does not conform to TIFF specifications.

Compression and Decompression

Image compression is a process of storing an image in a way that uses less disk space than is used by the uncompressed image. In developing the compression and decompression routines for the HP Image Library, the image team had to account for three factors:

- The quality of the displayed image
- The speed of decompressing and displaying the image
- The amount of compression required.

The quality of the displayed image partly depends on whether lossless or lossy compression is used. Lossless compression involves techniques that allow the image to be perfectly reconstructed. This method of compression stores the information about the repetitive patterns of pixels rather than storing every pixel. For example, for a sequence of 25 pixels each having a value of 92, lossless compression could store two items: one pixel of value 92, and the information that this pixel repeats 25 times.

Lossy compression is a method that uses different techniques to achieve even lower storage requirements than lossless compression. Lossy compression is recommended for photographic images. For the HP Image Library, lossy compression is based on discrete cosine transform (DCT) techniques. These are numerical techniques that transform complex color or grayscale information into less complicated information. Although the original color or grayscale values are lost, the image normally appears identical when it is decompressed and displayed.

Table I
File Types Supported by the HP Image Library

File Type	Reads	Writes	Typical File Contents	Version
TIFF Types				5.0, 6.0
Bitonal	Yes	Yes	Line art, text	
Grayscale	Yes	Yes†	Black and white photos	
Palette	Yes	Yes	Color photos and X screen dumps	
RGB	Yes	Yes	High-quality color photos	
YCbCr	Yes	Yes	High-quality color photos	
Compressed TIFF Types				5.0, 6.0
PackBits	Yes	Yes	Compression for the bitonal images	
LZW	Yes	Yes	General-purpose compression	
CCITT Group 3	Yes	Yes	Common fax images	
CCITT Group 4	Yes	Yes	Common fax images	
JPEG	Yes	Yes	High-quality color photos	
X Image Types				X11
Xwd	Yes	No	Pixmap image from Xwd (Z format)	
Xbm	Yes	No	Bitonal X bitmap images	
Xpm	Yes	Yes	Color ASCII X pixmap images	
JFIF	Yes	Yes	Color photos and continuous tone images	8-R8
GIF	Yes	No	Xv/Xgif network images, dialup services	87a
Starbase Pixmap	Yes	No	HP Starbase pixmap images	1

† Writing grayscale images is supported only in 8-bit format.

To address user concerns about disk space, the HP Image Library supports a full range of compression and decompression methods. Application programs can access the compression and decompression methods listed in Table II through the HP Image Library.

Except for JPEG and JFIF compression, all compression methods are lossless. Although JPEG is lossy, that loss is normally unnoticeable unless the image is compressed by more than a factor of 20 times.

Typically, lossless compression methods reduce the storage requirements to 50% of the original disk space. However, by

Table II
Compression and Decompression Methods Provided in the HP Image Library

Compression and Decompression Method	Image Types
JPEG	Grayscale, RGB, and YCbCr
JFIF	YCbCr
Group 3	Bitonal
CCITT Group 3	Bitonal
CCITT Group 4	Bitonal
LZW	Bitonal, grayscale, palette, RGB, and YCbCr
PackBits	Grayscale, palette, bitonal

using the lossy JPEG compression technique, a typical compression reduces the storage needed to 5% of the original disk space. However, the image library still displays the photograph with high-quality resolution and no noticeable change in display time.

All the compression methods provide fast display and high-quality image appearance. For JPEG, the fastest display implementation is provided in the version of the HP Image Library supplied with HP MPower. With JPEG, the programmer can choose the amount of compression by trading off the amount of compression with the image quality desired.

Before compressing a color image file with JPEG, the application can save additional space by first converting the file to YCbCr format and subsampling it. Subsampling is a technique that reduces the color information to be compressed, without noticeable change in the image quality. Subsampling is really a form of compression, because it stores fewer pixels than exist in the image. Because the human eye perceives degrees of brightness with much higher resolution than exact shades of color, only a fraction of the chrominance samples are stored. After subsampling, the subsampled bits can be replicated before displaying the image. The replication process is called upsampling. The upsampled image looks identical to the original image.

If the application needs to send files to a fax machine, it can use the CCITT Group 3 and Group 4 compression methods. However, most fax machines support only Group 3 format. The HP Image Library can convert any type of image it supports to any other type it supports. Therefore, a photographic image in RGB format can be converted to CCITT Group 3 format so that the file can be sent to a fax machine.

To compress images, end users must experiment with compression methods on various types of images. The following general guidelines are helpful in determining which compression method to use for reducing the storage requirements of images:

- For photographic images when some of the image detail can be sacrificed, choose JPEG (choosing the desired compression amount) or JFIF.
- For Xwd (X window) screen dumps, or other computer-generated images, use LZW.
- For image files being sent to a fax machine, use CCITT Group 3.
- For fax files to be stored or sent to HP MPower systems, use either CCITT Group 4 (if the content is mostly white space, such as text) or either Group 3 or CCITT Group 3 (if the content is highly detailed).
- For images being ported to various non-HP-UX systems, choose PackBits.

Display Quality

Because the resolution of an image on paper can be 20 to 100 times higher than that of the computer screen, end users are concerned about image display quality. To optimize the appearance of displayed images, the image team included the following capabilities in the HP Image Library:

- Simultaneous display of different color images by sharing a common color palette for 8-plane displays
- Gamma adjustment of colors, such as changing the brightness of an image
- Programmable choices for the type of dithering technique used
- Automatic dithering of images on 8-plane displays and remapping of pixel tones
- Automatic conversion of color images to grayscale format on monochrome displays
- Programmable conversion of bitonal images to display as grayscale images.

Dithering

Dithering is a technique that trades screen resolution for more colors or gray levels. While the resulting image may be more grainy than the undithered version, the contrast between the greater range of colors or gray levels provides improved appearance. Dithering is accomplished by modulating the color values between two adjacent color tones. From a moderate distance, the human eye automatically blends these regions of color together and perceives the average intensity.

Two options are available in the HP Image Library for dithering: error-diffusion dithering, using what is called the Floyd-Steinberg method, and area-based dithering. Compared to error-diffusion dithering, area-based dithering provides a more matted appearance in the image (see Fig. 4a on page 42). However, if speed is the primary concern, an image can be displayed faster by using area-based dithering.

For area-based dithering the HP Image Library performs the following two steps:

1. In one area at a time, it applies an 8-by-8 array of positive and negative values to the color or grayscale values. This step preserves the average color or grayscale level in the area, because the sum of the values in the array is always zero.

In a simplified example, this step might subtract 20 from the value of half the pixels and add 20 to the value of the other half of the pixels. Thus, the average value of the pixels in the area remains the same.

2. The values from step 1 are reduced to a number of values that the screen can display. For example, in a grayscale image values 1 through 256 need to become values 1 through 32 for a monochrome screen. In this example, each value is divided by 8 (256/32). Pixels of gray level 1 through 8 become value 1, values 9 through 16 become 2, and so on.

In error-diffusion dithering, the HP Image Library changes the values of pixels one at a time rather than working on areas of pixels. After assigning a dithered value to one pixel, error diffusion records what the difference was in that pixel's value. That difference is then added to the next pixel's value before it receives its dithered value.

For instance, suppose there are two neighboring pixels with values of 90 and 140. If the pixel with the value 90 receives a dithered value of 100, the difference is 10. So, 10 is added to the next pixel's value of 140, making a pixel of value 150 before it receives its dithered value. Perhaps the 150 is dithered to 170, making the current difference 20. So, 20 is added to the next pixel, and the process repeats.

Error diffusion is the dithering method used by the HP MPower ImageView tool,† whereas area-based dithering is the default dithering method.

Before displaying an image, the HP Image Library checks the characteristics of the display device to determine if dithering is necessary. The HP Image Library automatically dithers the image when displaying an RGB or YCbCr image on a pseudo color display device or displaying a grayscale image on a bitonal device.

For RGB images on a pseudo color device, the HP Image Library uses area-based dithering. For grayscale images on a bitonal device, the library uses error diffusion. When a grayscale image is displayed on a bitonal device, the HP Image Library dithers it to a bitonal image.

By default, the HP Image Library chooses the dithering method by using the following criteria:

- If the screen is a pseudo color (palette) image, area-based dithering is used. The image is first converted to RGB and then to palette.
- If the screen is monochrome (bitonal), error diffusion is used. The image is first converted to grayscale, then to bitonal.

Image Conversion for Space and Readability

For reasons such as saving space and improving the readability of text-oriented images, an application can convert images from one image type to any other image type.

Space and Color Images. The high quality of a 24-bit RGB image is only visible on a 24-plane system. When displayed on an 8-plane display system, an RGB image is automatically converted to palette. However, an uncompressed 24-bit RGB image occupies more space than it would as a palette image.

To save space on 8-plane display systems, the application can convert the 24-bit RGB images to palette or YCbCr format. YCbCr format is preferable if JPEG compression is also used. Beyond the additional compression possible, the image space required can also be reduced through the subsampling technique.

Space and Text Images. Concerning monochrome images, the programmer can save space by converting grayscale images of text to bitonal images. A grayscale image requires eight times the space required by a bitonal image.

† ImageView is a tool for displaying, manipulating, saving, and printing images of different image types.

HP Image Library Scaling Functions

The HP Image Library scaling capability performs three types of scaling according to which option is used: scale to gray, area-sample scaling, and simple scaling. In each case, the scaling algorithm accounts for the type of image involved, whether any image type conversion is needed, and whether the program is requesting that the image be scaled up or down.

Simple Scaling

Simple scaling gives the fastest scaling performance but the lowest image quality. This method uses pixel replication if the image is enlarged and pixel decimation if the image is being reduced. No image conversion is performed.

Area-Sample Scaling

Area-sample scaling gives the highest-quality scaling results. This method only applies to scaling an image down in the current release of the HP Image Library. In scaling the image down, sample scaling uses area sampling techniques based on the image type:

- For a color palette image, the image is first converted to an RGB image. The RGB image is scaled by area sampling.
- For bitonal images, the image is temporarily converted to an averaged grayscale type and then the image's pixels are set to on or off, based on a threshold value set by the application. (High-threshold values darken the image and low-threshold values lighten it.) The resulting grayscale image is then scaled by area sampling.

Bitonal-to-Gray Scaling

Bitonal-to-gray scaling applies only when scaling down bitonal images. It converts bitonal images to grayscale and uses area-sampling techniques to scale the image down. To convert a bitonal image to a grayscale image, the HP Image Library assigns each black or white pixel a gray level between 0 and 255.

While a scanned photographic image should be a grayscale image to retain the different levels of gray, a scanned text-oriented document or other text image can be bitonal and still be read from the screen by users.

Readability and Text Images. When a grayscale image of text is stored as bitonal, it can be converted back to grayscale if it was scaled down for display. The result is text that is easier to read. Thus, the image can be stored as a bitonal image, using less space, and displayed as a grayscale image.

Image Manipulation

To manipulate images, the HP Image Library includes a number of functions for scaling, rotating, mirroring, cropping, and changing image colors.

Scaling Images. The scaling function maps one image into an image of a different resolution. It allows the application to account for the differences in resolution between the display screen and the image captured by devices such as scanners. It also provides a mechanism to resize images to different window resolutions (see "HP Image Library Scaling Functions," above).

An application can scale an image up or down. For example, scanners are typically 300 dpi and display monitors are only 90 dpi. Therefore, images scanned at 100 dpi or higher resolution must be scaled down to the screen resolution.

The scaling function includes options for faster scaling or high-quality scaling. Faster scaling replicates or removes pixels, depending on whether the image is being scaled up or down. High-quality scaling converts bitonal images to

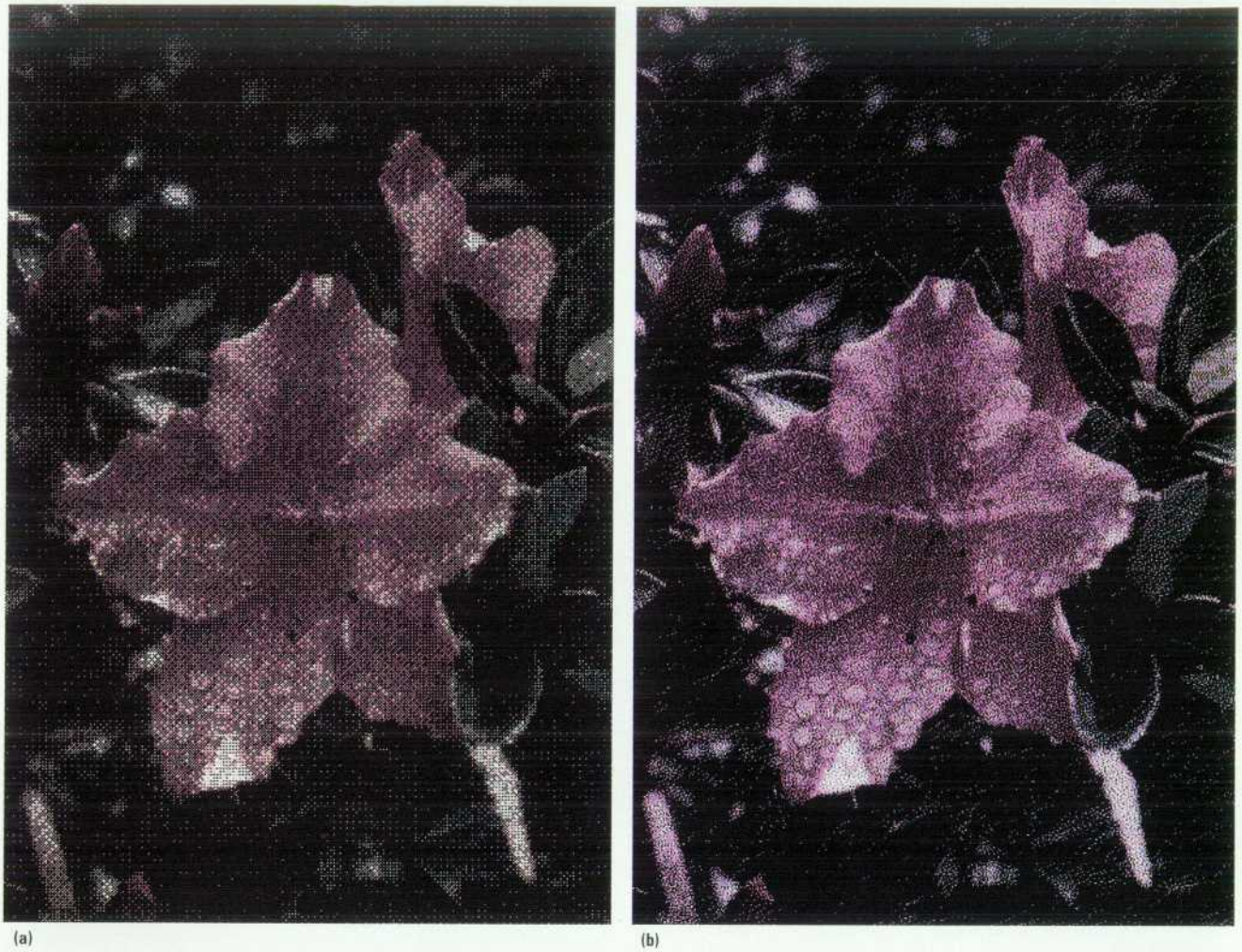


Fig. 4. A comparison between (a) area-based dithering and (b) error-diffusion dithering.

grayscale or scales color images by area sampling. Area sampling replicates pixels based on the average color values of an area of pixels, producing a clearer image.

Rotating and Mirroring Images. The rotation function rotates an image at integer multiples of 90 degrees. The rotation can be clockwise or counterclockwise. If the image is rotated by 90 or 270 degrees, the width and height of the image are reversed. Otherwise, the image retains the same dimensions as before rotation. The mirroring function mirrors the image about the x or y axis.

Cropping Images. The cropping function extracts a rectangular section of an image, creating a cropped image that is either the same size or smaller than the original image. An application might combine the scale and crop functions to implement features such as panning and zooming.

Changing Image Colors. The color mapping function changes the image colors by mapping the color values of the image pixels into different color values. This function can be used by applications to change the colors, brightness, or contrast of the image.

Custom File Types and Custom Functions

To allow programmers to extend the HP Image Library, functions are included for defining new types of files and creating custom functions.

Custom File Types. For unsupported types of image files, the programmer can extend the extensible file support library. The programmer defines a new type of file using an extensible file support function and then rebuilds the extensible file support library.

Thus, if an existing application accesses files through the extensible file support library, the newly defined file type can be accessed as just another extensible file support file. This object-oriented approach allows the programmer to create routines without worrying about what type of file is being manipulated.

Custom Image Functions. For capabilities not available in the HP Image Library, the application can define a new function. An application-defined function (or custom function) can be a producer, a filter, or a consumer function.

Fig. 5. An HP ImageView screen.

Application-defined producers and consumers might be used to read or write to an image-capable device. The application-defined producer outputs a pipe image, ideally in strips, that may have an input image type that is unknown to the image library but known to the application-defined consumer. The application-defined consumer can be defined to accept a standard or nonstandard input image type that is unknown to the HP Image Library.

An application-defined filter can be created to operate on a client image that is not in a format supported by the HP Image Library. For example, the client image can be created to support a color image in CMYK (cyan, magenta, yellow, and black) format. While standard HP Image Library functions can read and write this client image, operations to manipulate the image require an application-defined function.

End-User Image Tools

Based on the image and extensible file support libraries, end-user tools have been created by the image team and several other HP teams. For example, the HP online help facility, and the HP MPower components fax, DeskScan/UX, Whiteboard, ImageView, and HP SharedPrint use these libraries for image display and manipulation.

A central part of the HP MPower image display is ImageView, an OSF/Motif-based image display application (see Fig. 5). A basic version of this application is built into the standard run-time applications on HP 9000 Series 700 and 800 systems.

As a client application of the image libraries, ImageView displays all the file types supported by the extensible file support library. The user displays an image by double-clicking on a file icon in HP VUE. The user can then zoom in on arbitrary areas of specific interest and resize images by dragging a corner of the window.

In the HP MPower version of ImageView, users can also compress an image file, print images, adjust contrast, brightness, and orientation, and save those changes. Also, users can display images without dithering and fix the image scale during display.

Image Developer's Kit

The image team created an image developer's kit that programmers can use to create applications built on the HP Image Library. The applications developed by programmers can be clients of the X server, or they can work solely with image files without using X. Most applications access both image files and the X server because it is the only means for displaying images.

The toolkit can be installed on HP 9000 Series 700 systems and consists of the following components:

- Header files for the HP Image Library functions
- A range of sample image files, such as TIFF, GIF, and Xpm files
- Man pages that describe all HP Image Library function calls
- Source files for sample applications that contain calls to the HP Image Library functions
- Makefiles that convert sample source files into executable programs
- Source files that show how to extend the HP Image Library, adding support for other types of files
- A utility called imageutil that provides command line options for image viewing and manipulation.

Acknowledgments

The authors would like to thank the image library engineering team for their valuable contributions to the image project. Enrique Santos and Melinda Shebell led the two phases of the product development. Peter Kaczowka designed the overall architecture of the image library and the extended file support library. Derek Larsson created functions that manipulate, dither, and display images and ImageView enhancements in HP MPower and he helped with creating this article. Lynn Miller and Joel Stave developed the original ImageView application. Peter Voegelin developed compression and decompression routines, and Noelie Alito developed routines for reading and writing TIFF images. Daniel Bass developed compression and decompression routines including JPEG routines.

A Printing Solution for a Multimedia Environment

For environments in which users are confronted with a myriad of printers to choose from, HP SharedPrint provides a simple graphical interface that enables users to select a target printer and a set of options without encountering the typical problems associated with this process.

by John Mandler

Most computer users in a networked computer environment take certain aspects of the system for granted—especially shared resources such as disks and printers. Users expect these resources to work flawlessly without any concern or interference on their part. Unfortunately, the ideal is not always possible, especially for printers. Printers often fall far short of users' expectations. Because of this, we placed special emphasis on providing a robust, easy-to-use printing solution for HP MPower.

To understand the HP MPower printing solution it is first necessary to examine the fundamentals of a spooled printing system. Fig. 1 shows the phases and basic operations of a spooled printing system.

The user controls the specification phase by selecting an output device from a list provided by the system. The object (file) to print forms a second part of the the job specification. Finally, any options that control the appearance of the final output should be noted. The options may be used to control when the job is printed, to define the desired feedback, to select some feature of the output device, or to specify the final appearance of the printed output.

The queue and control phase is handled by the native spooling system. It packages the user's job specification, locates the target machine, establishes the communication path, and places the job in a queue. The spooling system eventually dequeues the job, invoking the functions that process the job, and directs the output to the printer. The spooling system also handles query and control requests from the user, providing the feedback required for a robust system.

The processing phase performs the real work in a printing subsystem. This phase is initiated by the spooler, which starts a process that connects a dequeued file to the process input and the printer data port to its output port. The processing phase performs the translation and formatting of the input data stream, applying options as appropriate, and sending out printer control functions to the selected printer when necessary.

Problems

Each of these three steps—specification, queue/control, and processing—has problems that can range from nuisance issues such as the printer is out of paper to a compete breakdown of the printing subsystem (see Fig. 2). Problems in the specification phase include the need for the user submitting a print job to spend time finding the names and capabilities of available printers and spending more time learning how to add special printer control options to a print request. The options give the user more control over the final look of the document.

Queue and control problems include minimal to no feedback on the state of a job or printer. Failures are often not reported to the user, giving the appearance that jobs just disappeared from the queue. Administering the spooling system can be a difficult task. The steps to add a printer are not obvious and modifying a printer's default behavior requires more knowledge about the printer and programming than the user may want to know.

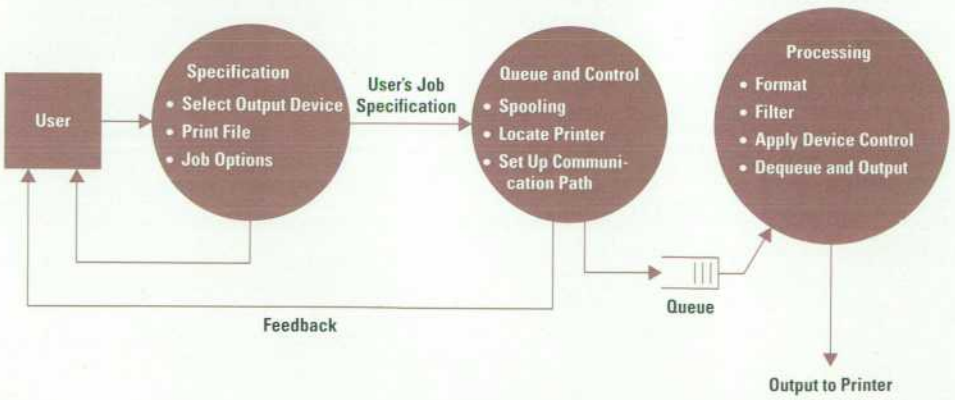


Fig. 1. Basic operation of a spooled printing system.

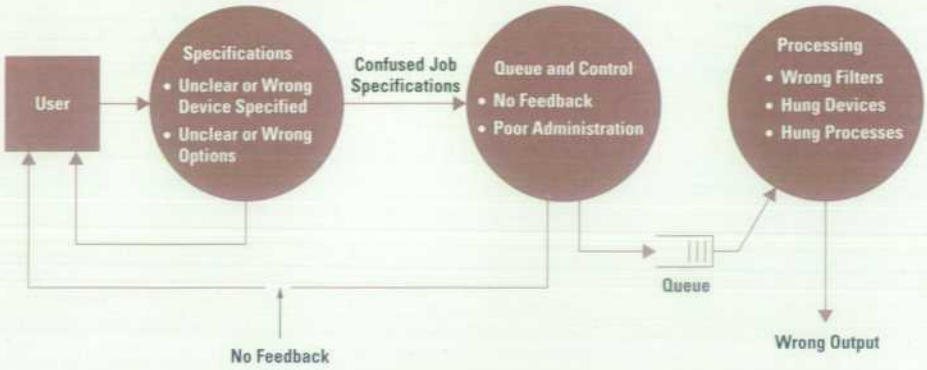


Fig. 2. Problems associated with some spooled printing systems.

The problems encountered during the processing phase can be the most difficult to diagnose and fix. Consider that the print job process is a background process running on a server, which is often remote from the user. If the wrong operations are performed on a print job, the result can be a hung spooling system, no output, or pages of useless output from the printer. Determining the failure requires intimate knowledge of the spooling system and any shell scripts or programs the spool system executes.

HP SharedPrint Solutions

The HP Mpower printing subsystem is based on the HP SharedPrint product. It solves many of the printing problems mentioned above by providing a robust, easy-to-use subsystem layered on top of a spooling system. The HP SharedPrint solution addresses the specification phase by providing a graphical user interface tool that includes a list of printers to choose from and a highlighted list of options that can be specified for a particular printer (see Fig. 3).

HP SharedPrint provides tools that enable the system administrator to set up a printer easily and quickly, making customization trivial. Another graphical user interface tool provides real-time status of print jobs and printers, along with control functions to cancel print jobs, add or delete printers, and modify a printer's default behavior (see Fig. 4). Both of these tools are described in detail later in this article.

The HP SharedPrint processing modules eliminate the major problems that cause system administrators and end users so much trouble. An intelligent server provides print job processing based on an algorithm that dynamically builds and executes the elements necessary to handle the job properly. Built-in error detection logic is used to provide a hard-copy message to the user detailing any problems in processing the job.

HP SharedPrint Overview

HP SharedPrint has a client-server architecture. The client components include the two graphical user interface screens

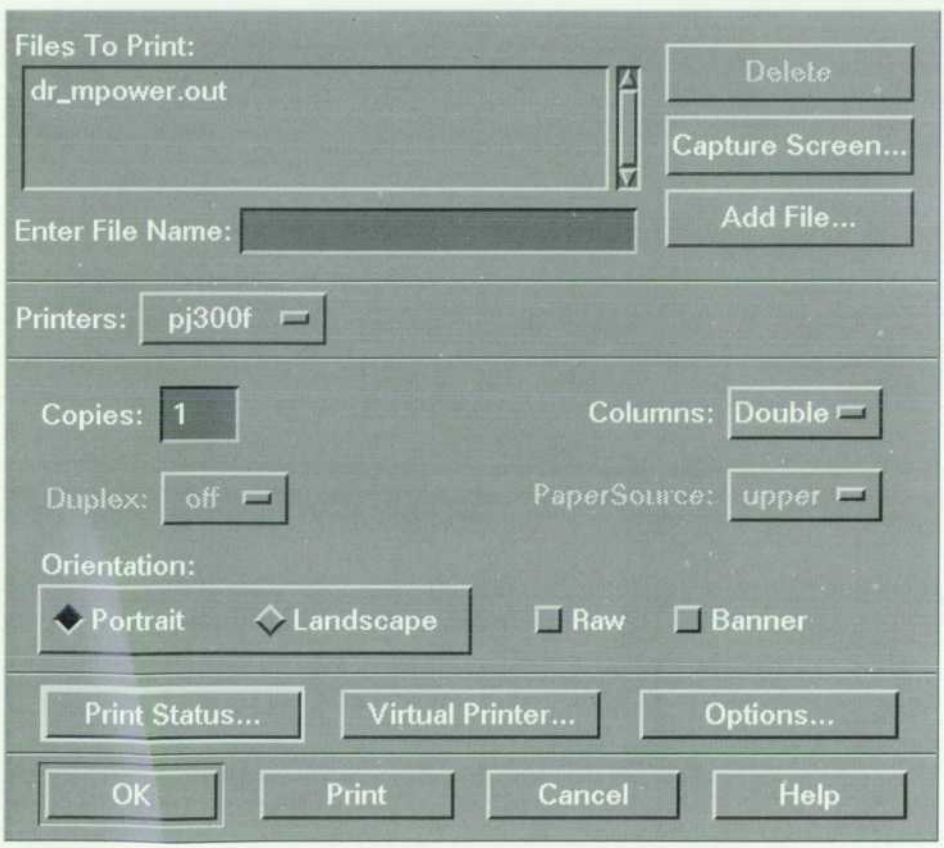


Fig. 3. HP SharedPrint graphical user interface showing a list of available printers.

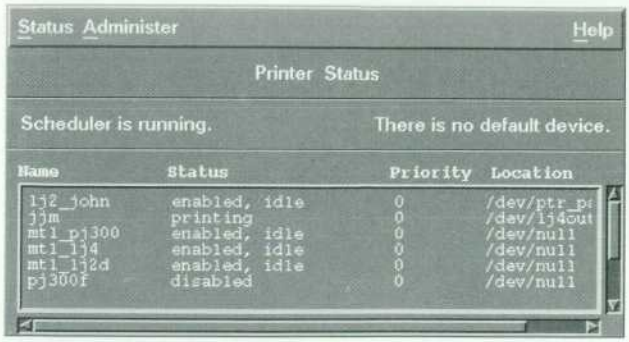


Fig. 4. HP SharedPrint's graphical user interface for system administration.

mentioned above. Fig. 5 shows a high-level view of the HP SharedPrint client/server system. The job specification client communicates with the HP SharedPrint server via the base spooling system. The server provides real-time feedback to the user about what options are valid for a specific job and printer combination.

One of the major features of HP SharedPrint is its ability to print many of the common file types† transparently on any of the supported printers. This functionality eliminates the need for the user to know which file types can be printed on which devices.

The HP SharedPrint server, being highly configurable, can be used to drive smart as well as dumb printers. Also, because of its highly modular design, the server can be layered on any spooling system.

Fig. 6 shows a detailed view of the main components that make up the HP SharedPrint client/server system. The HP SharedPrint client contains the graphical user interface programs sprint (HP SharedPrint print) and spadmin (HP SharedPrint administration). Sprint prompts the user for the file to print (① in Fig. 6) and the designated printer and then uses NCS (network computing services) to connect to the HP SharedPrint feedback server ② and ③. The feedback server hands sprint a set of valid options based on the user's inputs ④. The option information is used to validate which options the user can select for the current job.

After completing the job specification, the user selects the OK button in the sprint window to queue the job for printing. At this point the job request moves to the line printer spooler (lp) client ⑤. The lp client packages the print request and connects to the lpsched daemon located on the lp server ⑥. The lpsched daemon places the job request in the printer queue and at the appropriate time starts the HP SharedPrint job processor to process the job ⑦.

The HP SharedPrint server includes the HP SharedPrint job processor, configuration files that specify various mappings, filters, and drivers that perform translation and formatting, and the HP SharedPrint feedback server. Data from the configuration files is used by the feedback server and the job processor. The feedback server uses the data to provide option information to the sprint user interface, and the job processor uses the data to define the processing steps. The job processor invokes the filters and drivers to finish processing a job. The filters translate and format data files going to the selected printer, and the drivers place device-specific control information in the data stream going to the printer.

The Print Client

The sprint print client enhances the basic HP VUE printing model by providing an easy-to-use graphical interface. Sprint does not alter or replace the current HP-UX* spooling system functionality but is layered on top of the spooling system. This leads to a smooth transition from the standard HP-UX printing models to HP SharedPrint. The sprint graphical user interface allows users to:

- Select a file to print via a drag and drop operation from the HP VUE file manager interface
- Select a printer from a displayed list of printers
- Set options based on real-time feedback
- Save groups of options for reuse later on similar print jobs.

† A file type is a special data format required of all printers, such as PCL image files, RTL (raster transfer language), PostScript, or applications (image or text).

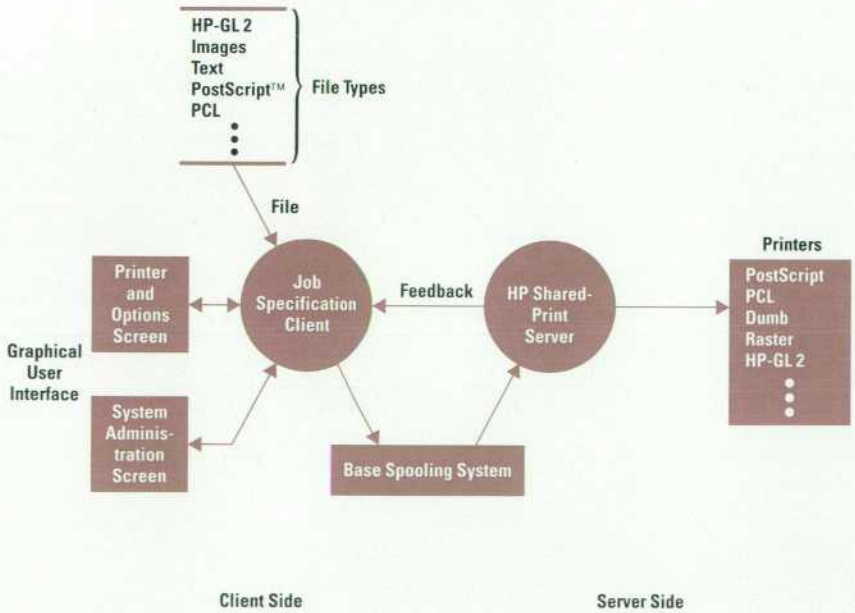


Fig. 5. A high-level view of HP SharedPrint's client/server architecture.

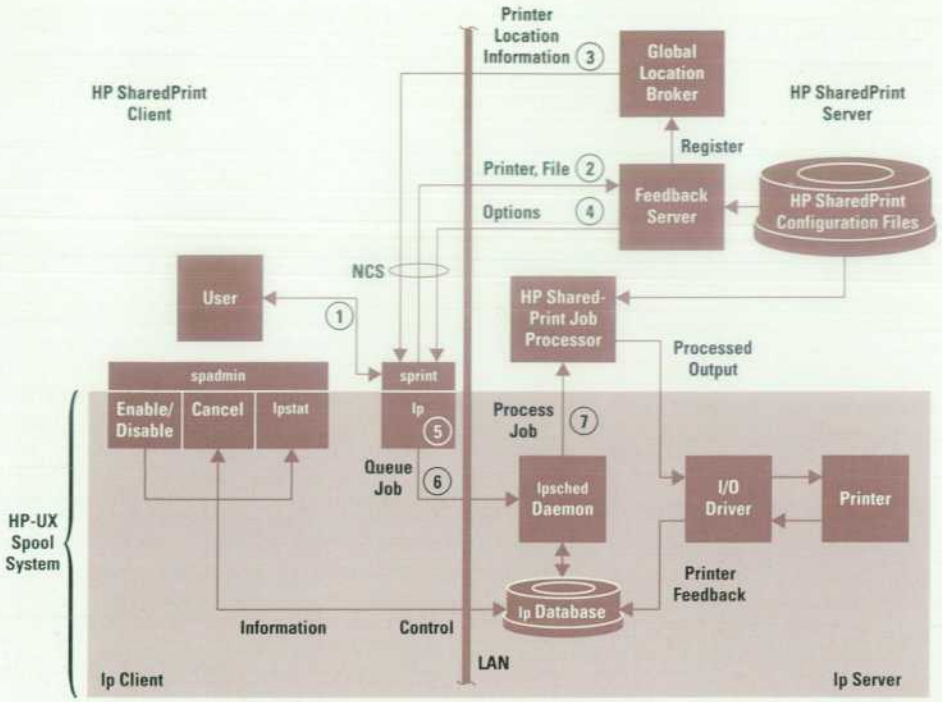


Fig. 6. A detailed block diagram showing the components that make up HP SharedPrint's client/server system.

The top level interface for sprint is shown in Fig. 3. The dimmed buttons represent unavailable items. For example, the menu item PaperSource is dimmed to indicate that the selected printer does not support multiple paper trays.

The button marked Options will take the user to the screen shown in Fig. 7. This screen provides another set of options. These options are sent to sprint by the feedback server.

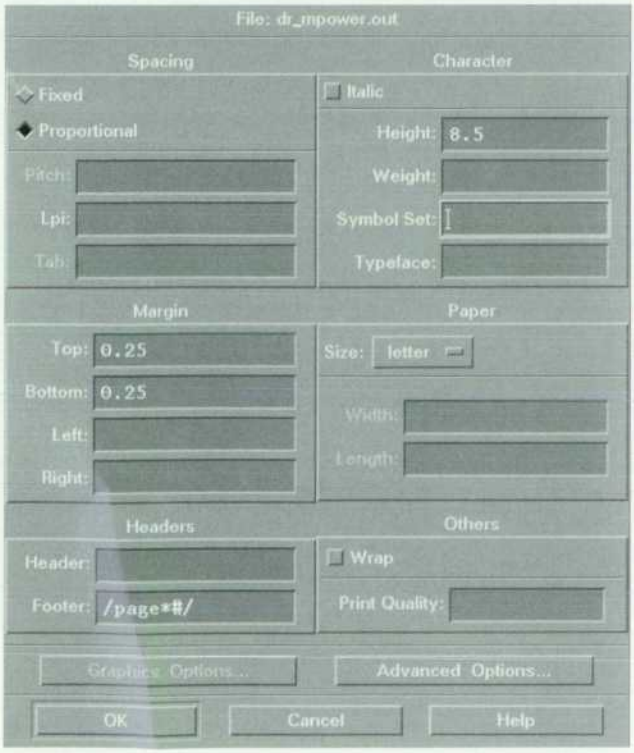


Fig. 7. The Options screen that shows the options available for a certain printer.

The Administration Client

The HP SharedPrint administration (spadmin) graphical user interface is shown in Fig. 4. Selecting the printer icon from the HP VUE front panel activates the spadmin client. Print request or printer status information is displayed and updated every 30 seconds. The Administer menu enables more complex tasks such as editing printer configuration files, adding and deleting printers, and making other changes to the print spooler state.

Core Server

The HP SharedPrint core server components are shown in Fig. 8. The core server performs the processing for a print job and provides the options feedback to the user. Two groups of configuration files are used for processing a job. One group contains information that defines job processing and feedback rules. The other group contains printer configuration information that defines the default settings and

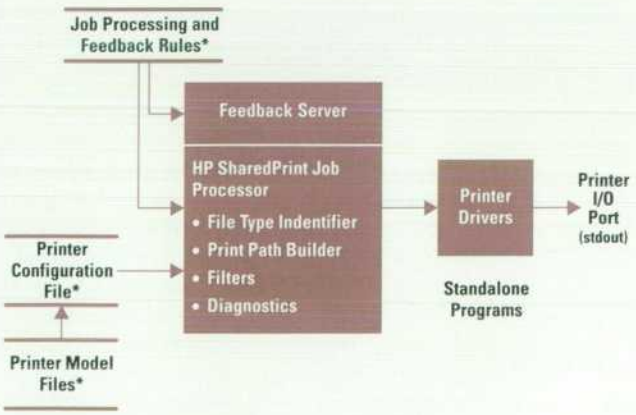


Fig. 8. HP SharedPrint's core server components.

behavior for a particular printer. There is one printer configuration file for each printer connected to the system. Filters and drivers are standalone programs invoked by the core server to process a print job.

The core server consists of two separate processes that share a common set of data and some common functions. These processes are the job processor and the feedback server. The job processor is invoked by the print daemon lpsched with the arguments:

```
job_id  user_name  title  copies  options  files
```

The first four arguments represent the job number, the user who queued the job, the title of the job, and the number of copies to print. The options argument contains zero or more tokens, each representing an option specified by the user queueing the job. For example –charheight 8 –orientation landscape are typical tokens in the options argument. These options are set in the sprint client or on the lp command line by prepending the letter –o to the option string. The files argument is a list of one or more files to be printed.

The job processor serves all the supported printers by dynamically configuring itself for each print job. The information needed for this operation comes from the printer configuration file. Each printer requires a unique printer configuration file, which is created when a printer is added to the system. This configuration file can be modified at any time via the HP SharedPrint client spadmin or by using any text editor.

The job processor uses some control programs to read the configuration files to determine which set of filters to execute to print a job properly. Details of these steps are described later.

The other process in the core server, the feedback server, is run on every system where an HP SharedPrint printer is installed. This server provides the sprint interface with feedback on what options are valid for a specific file and printer combination.

The feedback server is automatically started when the first HP SharedPrint printer is added to the system. Other HP SharedPrint printers on the same system use the same HP SharedPrint server.

Configuration Files. The configuration files shown in Fig. 8 are text files that provide all the information necessary to process any of the supported file types on any of the supported printers. Users can modify any of these files using an editor, or in the case of the printer configuration file, the HP SharedPrint system administration utility spadmin. The configuration files include:

- Printer model files. These files contain information for specific printer models. For any particular printer model, this file includes the options supported, default values for any filters, and the file types the printer model can handle.
- Printer configuration file. This file contains information about a specific printer. The file is created from one of the printer model files when a printer is added to the system. Users can modify this file to reflect changes made to the printer, such as swapping paper trays or inserting font cartridges. Changes take effect at the next print job.
- Object name extension file. This file contains a list of entries that map file extension values to file types. For example, files with the extension .c (C source files) would be mapped to an ASCII file type.
- Pipeline file. This configuration file specifies a filter pipeline process for each combination of input and output file type. Each of the filter specifications can include options and white space.
- Options map file. This file lists supported options for each filter or driver.
- Options file. This file lists known options and aliases.
- Types file. This file lists known file types and aliases.

These last five configuration files make up the files labeled as job processing and feedback rules in Fig. 8.

Filters. Filters are programs that transform a data file from one format to another. Filters perform either translation or formatting. Translators change the encoding of the information in the data file without changing its appearance. For example, in converting ASCII text to PostScript format the text remains the same, but the file is expanded to contain PostScript commands. Formatters modify how the information in the data file will be rendered on the page. For example, a formatter might rotate or scale an image, add commands for multicolumn text printing, or add footers and headers to a document. These filters often have options that the user can set to control the final appearance.

Filters are invoked by the job processor. Each filter reads from standard input and writes to standard output with error messages being sent to standard error. The filters supported by HP SharedPrint include:

- txpcl. This filter formats and translates text documents into PCL (printer control language) format. The formatting options allow users to change point size, print portrait or landscape, perform double column printing, add headers and footers, and change the typeface and symbol-set mapping.
- txps. This filter formats and translates text to PostScript. It also supports most of the options included in the text-to-PCL filter txpcl.
- cgmhpgl2. This filter reads binary CGM (computer graphics metafile) format and produces an HP-GL 2 byte stream. It is used to obtain 2D graphics output from StarBase or HP-PHIGS CGM files.
- psrip. This filter allows users to print a PostScript file on a non-PostScript printer. The filter reads a level-1 PostScript file and creates a raster image for each page. The raster image is fed to a device-specific formatter which adds the appropriate control or command codes for the target device.
- pclrip. This filter performs the same function as the psrip filter on PCL files. The PCL level can be level 1 through 4 and can include the level-5 PCL scalable typefaces. PCL 3+, used by the HP PaintJet, PaintJet XL and DeskJet 500C printers, is not supported by this translator. The HP-GL 2 extension to level-5 PCL is also not supported by this translator.
- ilFilter. This is an image library filter that converts all HP MPower supported image formats, a level-1 PostScript file, or a PCL 4 file into a Postscript, RTL, or PCL file. The filter is composed of three stages: a producer that converts the input object into an image library format, one or more filters that

manipulate the raster image, and a consumer that converts the image library output to a PostScript, PCL, or RTL file.

Drivers. The function of a driver is to add job control information to the data stream produced by the filters. It does not interpret or change the data stream, but adds printer-specific commands to the byte stream. Think of the driver as handling printer-specific functions, whereas filters handle language-specific functions.

The job control information depends on the target printer. Each job control feature in the printer can be selected or set by the system administrator when the printer is configured or specified by users in their personal options files.

In most cases the driver will send a header with any job control information then cat the input stream from the filters to the output stream. In some cases, the driver will have to examine the first few bytes of a file to see if there are any reset control characters that might interfere with the driver header stream. For instance, PCL files may contain an ESC E that resets the printer state. The driver will have to strip these bytes if a header must be sent.

Processing Steps

For each print job, the HP SharedPrint job processor executes the algorithm outlined in Fig. 9. Each of the steps in Fig. 9 represents a module or inline code involved in processing a job.

Parse Command Line. This step is the interface to the base spooling system. Each spooling system passes the job data to the processing modules in a well-defined form. For the HP-UX spooling system, the job processing module is called via the lpsched daemon with the six command line arguments described earlier. The command line information, which includes the user name, printer name, job options, and files to print is collected and stored in the job processor for use as appropriate.

The main purpose of this module is to enclose the specifics of a spooling system in one location so that the process of porting HP SharedPrint to a new spooling system would involve changing only this module.

Get Printer Type. A key step in properly processing a job is to know the characteristics of the target device. One such printer characteristic is the type of languages or file types the printer understands.

The printer file types are listed in the printer configuration file, which is created when a printer is added to the spooling system. This step involves parsing the configuration file for the target printer. If no file-type entry exists in the configuration file, or the configuration file cannot be located, an error page is sent to the printer, the printer is disabled, and a mail message is sent to the user who queued the job.

Build Options List. Each job includes a list of options and associated values. The options list is built from default values defined in the printer configuration file, values passed by the user, or values added by the filters. User-specified options override options defined in the printer configuration file or the filters.

User-specified options can be passed from sprint via the lp command line. Since the HP SharedPrint option names are

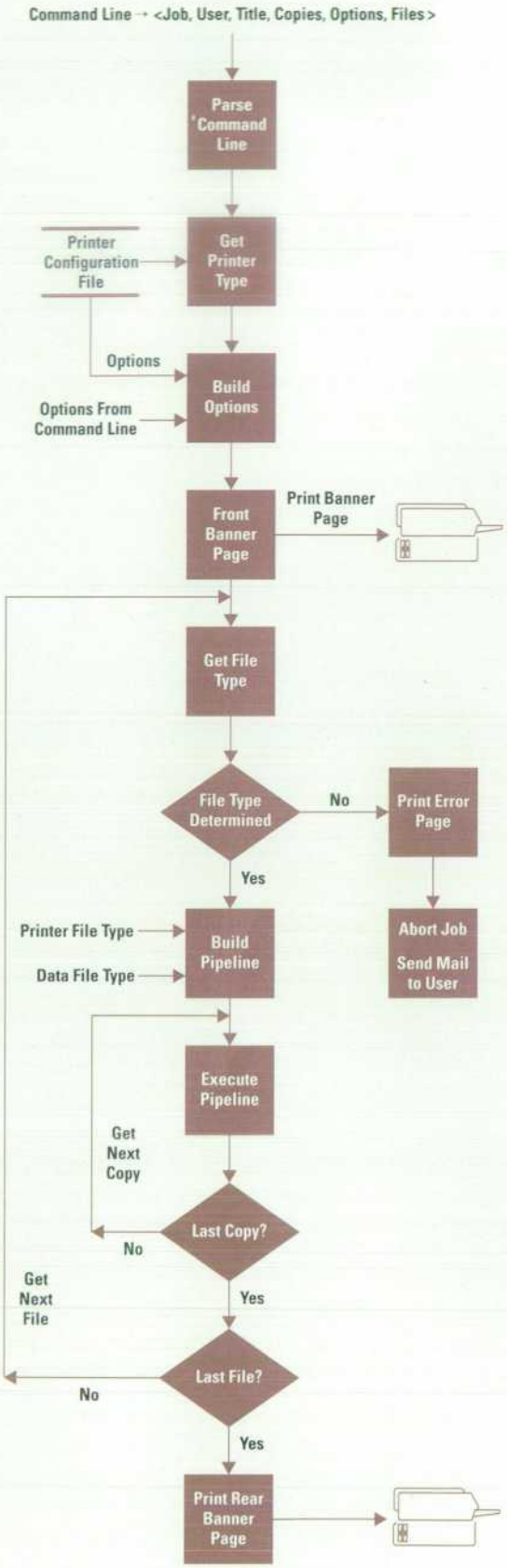


Fig. 9. Job processing algorithm.

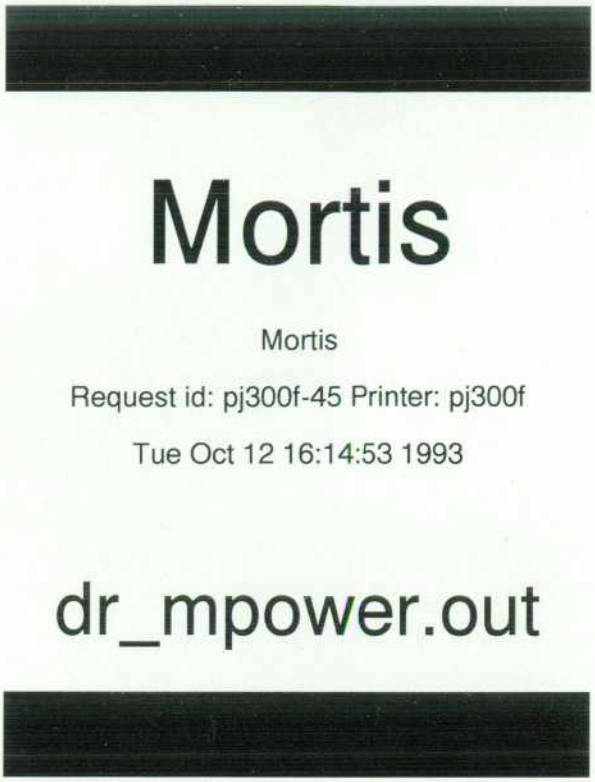


Fig. 10. PostScript banner page.

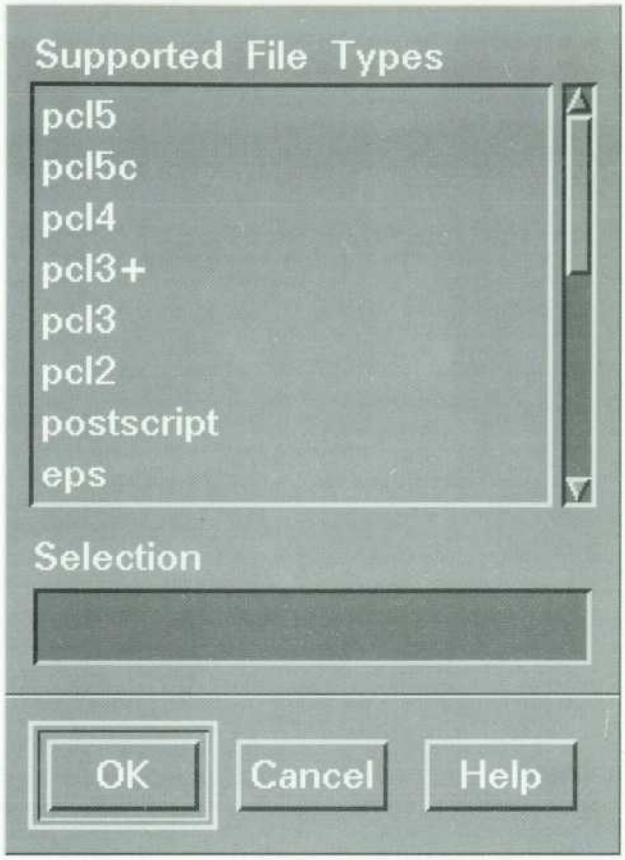


Fig. 11. The sprint dialog box asking about file type.

full-length strings, a set of aliases, which are defined in the options configuration file, can be used on the lp command line. The build-options-list operation will expand the aliased strings when the options list is built.

Front Banner Page. This step involves building and sending a banner page to the printer. This function creates a banner page file, then recursively calls the job processor to send it to the printer. In this way, any type of banner page can be sent to any printer. HP SharedPrint includes a banner page program for PostScript, PCL 5, and ASCII text. Fig. 10 is an example of the PostScript banner page.

Get File Type. This step determines the spooled file's file type. The job processor must know the file type of a job to determine if any processing is necessary to format the spooled file for the target printer. For instance, a PostScript file must be converted to PCL to print on an HP LaserJet printer. The file-typing process uses the following three methods (in the order given) to determine the file type for a job:

- Command line switch
- File type reader
- File extension value.

When a file type is found from any of these methods, the file-typing process terminates. If the file type cannot be determined from any of the above methods, an error page is printed, and the job is aborted.

- Command line switch. With this method if the user includes the file-type option flag (–file_type) on the command line, the file-type value will be set to the value following the –file_type flag. In this case HP SharedPrint assumes the user has some knowledge of the file's content that HP SharedPrint cannot determine. The sprint program will also prompt the user for the file type if the job processor cannot determine the file type. In this case sprint will pop up the dialog box shown in Fig. 11, asking the user to select one of the values.
- File type reader. This method compares the spooled file's contents with a predefined set of patterns for specific file types. These patterns are coded as C programs or Korn shell scripts. Each one of these programs or scripts expects a file as input. If the file type matches the defined pattern, the program or script writes the type string to standard output and returns an exit code of 0 to the calling routine. If there is no match, the program or script returns an exit code of 1.

  A special file-type program, based on the image library, is used for identifying image files. The image library includes a set of functions that allow programmers to open a file and determine if it is a file type that is supported by the image library. If the image library can process the file, the image file-type program writes the string "bitmap" to standard output and exits with a status code of 0.

  The module that determines a job's file type executes each of the file-reading programs and scripts located in a directory called filetypes until a match is found. If no match is found after all the programs have been invoked, the file extension mechanism is used to determine the file type.
- File extension value. File extensions are used by HP VUE to map file types to actions. For example, the extension .au is associated with the audio server, a process that plays or records audio files. HP SharedPrint uses many of the same extension values, mapping them to a file type in the object name extension file. The extension value is extracted from

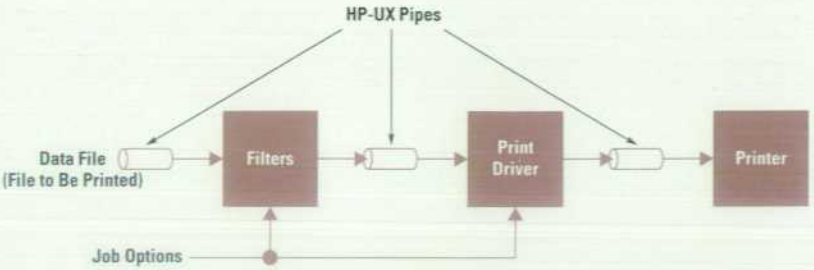


Fig. 12. Pipeline stages created during the pipeline building step in the job processing phase.

the leaf name of the original file that is passed by sprint in the lp spooler title option.

Build Pipeline. The processing pipeline performs the actual work of getting a file printed on the target device. The pipeline consists of one or more filters and a driver, with the output of each stage piped into the input of the next stage as shown in Fig. 12.

The pipeline is derived from the printer file type and the data file type. The pipeline builder scans the pipeline configuration file looking for file type matches. For each match, one or more filters is specified to perform the translation. If no match is found for the specified file types, an error page is printed and the job is aborted.

The pipeline is terminated by adding a driver module to the filter pipeline. The driver is determined from the printer configuration file. The pipeline string at this point will have the format: filter1 | filter2 | driver. Note that each filter might be a filter name followed by some options that control how the filter is executed.

The pipeline is completed by adding the appropriate options to each filter and the driver. The filter options file maps options that apply to each filter or driver so that the correct arguments are passed to the filters. The final pipeline string might look like:

```
txpcl -charheight 8 -orientation landscape | lj3.sh -copies 1
```

Execute Pipeline. The pipeline created in the build-pipeline stage is now executed. The output from the last stage of the pipeline is sent to standard output, which will be the I/O connection to the printer.

If any errors occur in the pipeline execution phase, they are collected and placed on an error page, which is printed at the end of the job. The error page function is called when some unrecoverable event occurs. A special log file contains all the output from the print script. Each filter or driver writes all error or warning messages to standard error. The job processor redirects this text to the error log.

At the completion of the job, the job processor scans the error log. If any information or messages are found in the error log, the job processor builds a text page containing the error messages and some debug information and then prints the text page on the printer.

Diagnostic information is included in the error page to aid in the analysis of the problem.

Final Steps. The last-copy step executes the pipeline again if another copy is needed, and the last-file step cycles the script back through processing the next file if there are multiple files to print. When recycling is required, the initial state of the options, printer configuration file, and output type remain the same and do not need to be recalculated. Finally, the rear banner step, which is identical to the front banner page, may be used to print just a short trailer, or in the case of printers that print face up, the banner page should be printed here.

Feedback Server

The feedback server helps users specify job parameters that optimize the hard-copy output for a given job. Without this feedback, users can be easily overwhelmed by the large number of available options. By enabling only those options that apply to the current job, the feedback server guides the user through the job specification process.

The feedback process shown in Fig. 13 begins with the user selecting a file to print on a specific printer. The sprint program uses the network computing services (NCS) to connect to the server and ask for a list of valid options. The feedback server executes the algorithm shown in Fig. 13 and returns a list of valid options to the client. The sprint program then enables the user to select only the options returned by the feedback server, disallowing (by dimming selections on the menu) other options to the user.

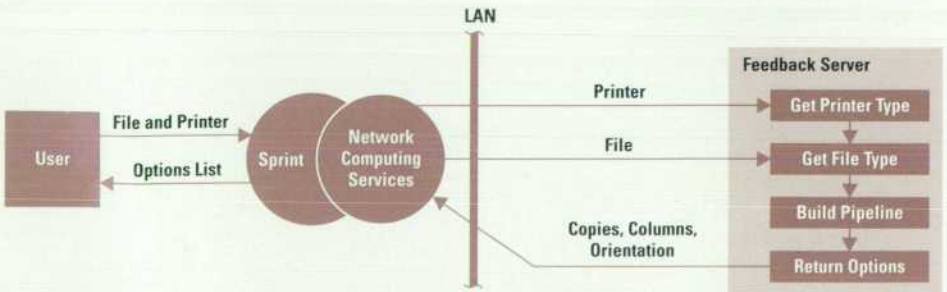


Fig. 13. The feedback server action and data flow.

HP SharedPrint Options

The options that apply to a given print job are a function of not only the target printer, but also any filters that may process the job. In the simplest situation, a file whose type matches the printer type can be sent without any filtering. In this case the set of valid options will be a function of the physical features of the printer, such as duplex or paper tray selection.

Even in the case in which the file type matches the printer, filters can be inserted to perform some preprocessing or formatting. For example, a PostScript document can be passed through a filter that places more than one logical page on a physical page.

HP SharedPrint precomputes the filters that will be used to process a job. This information is then used to extract the supported options from one of the files containing the job processing and feedback rules. It is this set of options that the user sees in the graphical user interface.

Along with each option is the value attached to that option. The filter processing the job must be fed both an option and a value. The source of the value comes from one of three locations: the user, the printer configuration file, or the filter, in descending order. For example, if the user specifies an option and value, this pair is passed to the filter. Otherwise if a value exists in the configuration file, it is used. Finally, if an option and value are not passed to a filter, the filter uses a built-in default value for the option. In this way, the behavior of a filter is controlled first by the user, then the system administrator (via the configuration file), and finally the programmer.

One problem with options is that there is no unified naming scheme. Two filters may provide the same functionality and use different strings to denote an option. HP SharedPrint addresses this by providing the options file to alias filter option names to a set of names defined by HP SharedPrint.

Conclusion

HP SharedPrint provides HP MPower with a robust printing solution. Users can drag and drop any HP MPower object to any printer and use the HP SharedPrint graphical user interface to control the printing process. The graphical user interface decreases user frustration, providing an easy-to-use printing interface. The intelligent server lowers the system administration burden by eliminating catastrophic failures and providing the flexibility to customize a printer's behavior.

Faxing Documents in HP MPower

The ability to transmit documents via standard telephone lines is greatly enhanced with the HP MPower fax utility which provides automatic dialing, transmission, and delivery of fax documents from a workstation.

by Francis P. Sung and Mark A. Johnson

Over the past ten years a revolution has occurred in telecommunications that has resulted in the capability to transmit documents easily and inexpensively using standard telephone communication channels. The fax machine has become pervasive on a global scale and is now a required tool for collaboration by all types of businesses. This success can be attributed to the standardization of fax communication and the introduction of low-cost, easy-to-use fax machines.

HP MPower fax provides a new level of refinement and capability in fax communication. HP MPower fax provides the following advantages and features:

- Client/server technology allows each HP MPower fax user access to fax capabilities at each user's desktop while sharing a fax modem at a remote location. This results in lower cost per user compared to multiple fax machines and more convenient access compared to sharing a single fax machine.
- Online documents can be conveniently faxed by dragging and dropping the document on the fax icon in the HP MPower user interface. This eliminates the process of printing and scanning the document into a fax machine. Document quality improves significantly since quality degradation occurs during the printing and scanning process.
- The problem of waiting for access to a fax machine is eliminated. Faxes sent from HP MPower fax are queued before being transmitted. If HP MPower fax is busy sending or receiving another fax, the fax waits in the queue and is automatically sent at a later time. Also, if the destination fax machine is busy HP MPower fax will automatically retry the fax transmission at a later time.
- Paper use is eliminated. Received faxes can be archived and viewed without printing. Online documents can be faxed directly without printing.
- Reliability is greatly improved since fax modems are inherently more reliable than fax machines.
- Costs are reduced by providing the capability to queue outgoing faxes for transmission when telecommunication rates are lower. Also, faxes that are being sent to the same number can be automatically batched together, saving the typically higher rate associated with the first minute of transmission.
- A palette of custom fax cover sheets can be conveniently created and accessed. Each custom cover sheet can include custom images such as corporate logos as well as fax sender information such as name, company, department, or any other information the user requires.
- A detailed log of all fax transmissions and receptions is kept. This can be used to monitor telecommunication charges and generate account billing information.
- Security of fax access is greatly enhanced. Different classes of HP MPower fax users can be created with varying privilege and security levels. Some classes could have the ability to send faxes to international fax numbers at any time of the day, while other classes might be restricted to local fax numbers and transmission times when telecommunication rates are discounted.
- Perhaps the biggest problem with a shared fax machine is the inability to route incoming faxes directly to the intended recipient. Incoming faxes can be viewed by anyone standing near the fax machine and are easily misplaced or lost. To avoid this problem, HP MPower fax can route incoming faxes directly to the recipient using a bar code on the cover sheet or the origination number of the fax.

HP MPower Fax Configuration

HP MPower fax uses a client/server configuration. Common functionality used by all HP MPower fax users resides on the fax server. Interaction with each fax user occurs at the fax client. Multiple fax clients are connected to a single fax server via a TCP/IP network connection. A fax client may also coexist on the same machine as a fax server. Up to 16 fax modems can be connected to a single fax server. Each fax modem supports a connection to one phone line. Fig. 1 shows a typical configuration in which HP MPower fax operates.

Major features at a fax client include an OSF/Motif graphical user interface, archival of received faxes, and a user-created directory services database of possible fax recipients. Major features offered by the fax server include fax modem communication, rendering, queuing and scheduling of outgoing faxes, temporary storage and routing of received faxes, and extensive databases for controlling server configuration, customization, and automation.

The process for transmitting a fax begins with the user selecting the file† to be faxed from the graphical user interface at the fax client. The file is then transferred to the fax server where it is combined with a cover sheet and rendered into the correct TIFF Class F format. The fax server then schedules the fax for transmission and stores it in a queue. When the appropriate time comes for transmission, communication with the fax modem begins and the fax is moved from the server queue to the fax modem and over the telephone line.

† Multiple files can also be sent to the fax server at the same time.

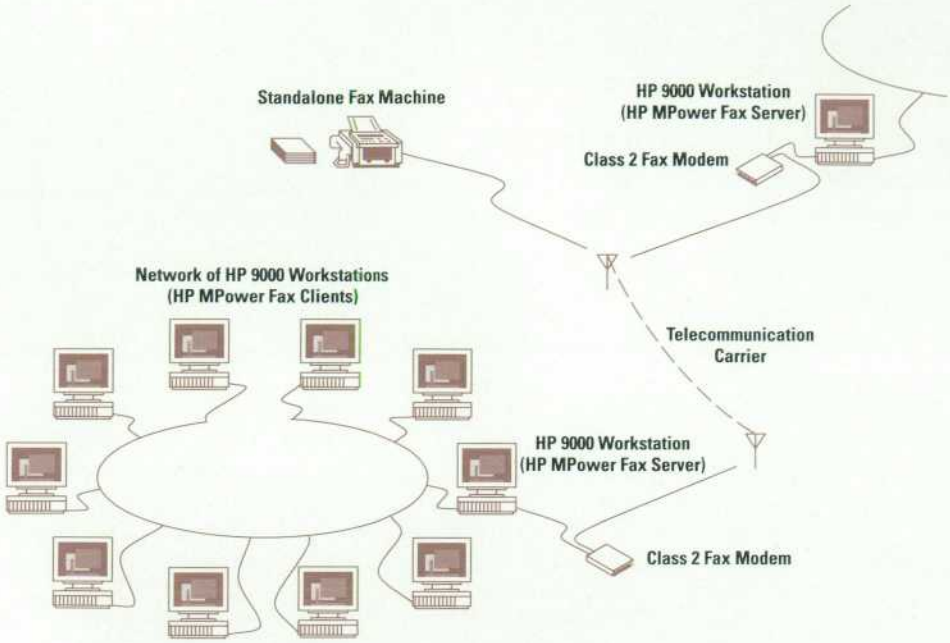


Fig. 1. A typical network configuration in which HP MPower fax operates.

When a fax is received, the fax server at the receiving end initiates communication with the fax modem and stores the incoming fax in temporary storage. If the fax can be routed, it is transferred to the appropriate fax client and the user is notified. Otherwise, the fax is stored in a general-delivery location on the fax server and all fax users are notified.

The process of transmitting a fax from a client and receiving a fax at a server is described in more detail in the following sections.

Fax Client

The primary component of the fax client is an OSF/Motif graphical user interface that allows sophisticated fax users to use the advanced functionality of the fax server while also enabling occasional users to send and receive faxes easily. The fax client's graphical user interface is integrated with HP VUE and other HP MPower components so that tasks such as dragging and dropping files to be faxed, printing received faxes, and scanning documents for inclusion in faxes are easily accomplished.

The user initiates a fax transmission by selecting the fax icon located on the HP VUE front panel or the HP MPower media panel. This causes the Compose Outgoing Fax window to appear (see Fig. 2). This is the primary window for creating, viewing, and sending faxes. An HP MPower fax user can send a fax by simply entering the fax number, dragging and dropping the fax attachments (files to be sent), and then selecting the Send Fax button. ASCII, PCL, PostScript,™ and EFS† files are automatically detected so the user does not need to know the file types of the attachments. The user can then select the Fax Status... button to monitor progress of the fax as it is being transmitted. The complete fax with cover sheet and attachments can be viewed before sending by selecting the View Fax... button. Since the cover sheets and rendering†† capability reside on the fax server, the fax client transfers the attachments to the fax server for rendering and then moves the rendered fax back to the fax client for viewing.

Advanced features in the Compose Outgoing Fax window are configurable so that sophisticated users can easily use advanced features without compromising ease of use for occasional fax users. The default configuration for the compose screen is a simple interface in which the user can gradually expose and use advanced features as they are learned.

Advanced features include a directory service database, email confirmation of transmitted faxes, handwritten signature placement on a fax cover sheet, automatic fax transmission at a specified later time (to lower telecommunication costs), automatic printing of transmitted faxes, bar code generation, and selection of optional styles and classes. The style of a fax determines the cover sheet appearance and physical size of the fax. Class determines the priority and parameters

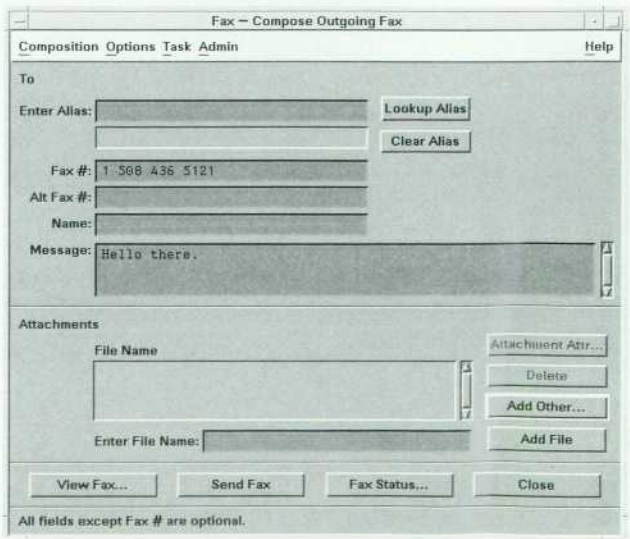


Fig. 2. The HP MPower fax window for composing fax messages.

† EFS, or extensible file support, is a part of the HP Image Library. EFS and the HP Image Library are described in the article on page 37.

†† Rendering is any form of drawing operation, including text, line, and raster output.

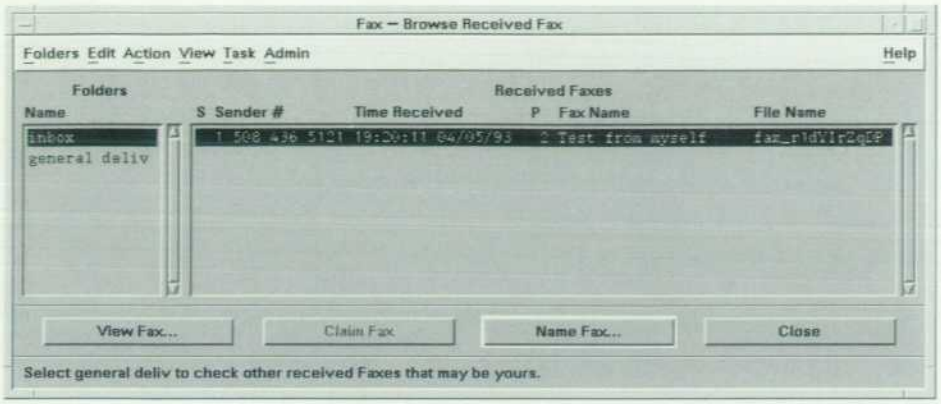


Fig. 3. The HP MPower fax window for browsing received faxes.

for scheduling and transmitting the fax. A directory service database can be created and manipulated on the client that contains the fax number, an alias name, and other pertinent information about frequent fax recipients. After creating an alias entry, the user only has to enter the alias of a fax recipient, which will cause the fax number and other pertinent data to be automatically accessed from the database and placed on the fax cover sheet. Multiple fax recipients can be grouped together into a group alias and saved in the directory service database. The user can enter the group alias name to send a single fax to multiple destinations at once.

In addition to file data, two other types of fax attachments can be generated from the Compose Outgoing Fax screen. A document scanned in from a scanner attached to the client's machine can be attached as part of an outgoing fax. Also, a snapshot of any currently displayed window can be captured and included as part of a fax.

Received faxes are manipulated from the Browse Received Fax screen (Fig. 3). Faxes that have been received can be annotated, printed, replied to, and archived in user-created folders. Typically, a fax is received by the fax server and placed in a general-delivery area. All fax users are then notified by a change in the appearance of the fax icon symbol located on the HP VUE front panel. When the user selects this icon, the Browse Received Fax screen will appear instead of the Compose Outgoing Fax screen. The cover sheet of any fax in general delivery can be viewed from this screen. After viewing the cover sheet, a user can claim a fax that appears to be that user's own. Claiming a fax moves the fax from general delivery on the fax server to an in-box on the fax client where the entire fax can be viewed from the Browse Received Fax screen. For installations in which confidentiality is very important, access to general delivery can be limited to a single individual user or group. This privileged user or group can view the cover sheet of each fax in general delivery and route it to the appropriate in-box.

Received faxes that have a bar-code symbol on the cover sheet can be routed directly to the in-box of the recipients. The fax server has a bar-code symbol decoder that scans all incoming fax cover sheets for a bar code. If a user's bar-code symbol is detected, the fax is routed directly to that user's in-box without going through general delivery. A bar-code symbol can be generated on the cover sheet of a fax transmitted by HP MPower fax for reception and routing by other HP MPower fax servers. For reception and routing of faxes originating from fax machines, a return cover sheet with the user's bar-code symbol can be generated. This return cover sheet is transmitted as the last sheet of a fax destined for the fax machine. The user at the fax machine can then use this return cover sheet as a fax cover sheet when replying. This allows faxes originating from fax machines to be routed when received by the HP MPower fax server. HP MPower fax can be set up to route all faxes that originate from a specified fax number to go to a specified fax user. This is useful if all faxes originating from a certain fax number are always destined to the same user.

Fax Server

The function of the fax server is to allow access to the fax resource from multiple fax clients simultaneously, while enforcing proper access control, accepting requests for fax transmission and scheduling, and receiving and disposing of incoming fax transmissions appropriately. To provide these services, the fax server is implemented as several programs that are run as separate processes. These processes maintain a set of databases and control some private directories and files. This complexity provides many of the advanced features offered by the fax server. A default configuration is provided with the product that allows any fax client to send and retrieve fax transmissions immediately after installation. With the default configuration, HP MPower fax is as easy to use as a standalone fax machine.

Fax Server Architecture

The HP MPower fax server is divided into three major functional components: the client-connection component, the fax-transmission component, and the fax-reception component. Eleven databases control the behavior of these three components. Fig. 4 shows the major components and databases of the fax server.

The circles in Fig. 4 represent programs, some of which are run as individual processes while others are invoked by the running processes as required. The parallel lines represent databases or storage areas.

The fax server components interact with the information in the databases to handle transmission and reception of faxes and communication with clients. The next few sections will discuss the databases shown in Fig. 4, and following that, the major functional components of the fax server will be described.

Fax Databases

The eleven fax server databases provide flexibility, configurability, expandability, convenience, and advanced features

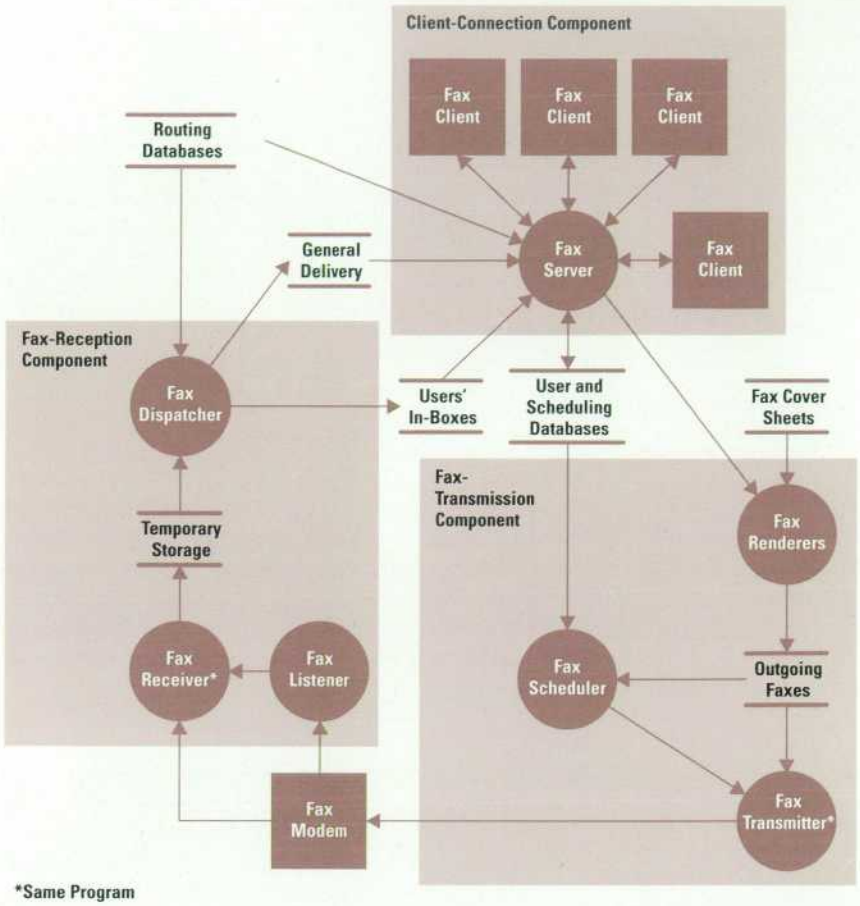
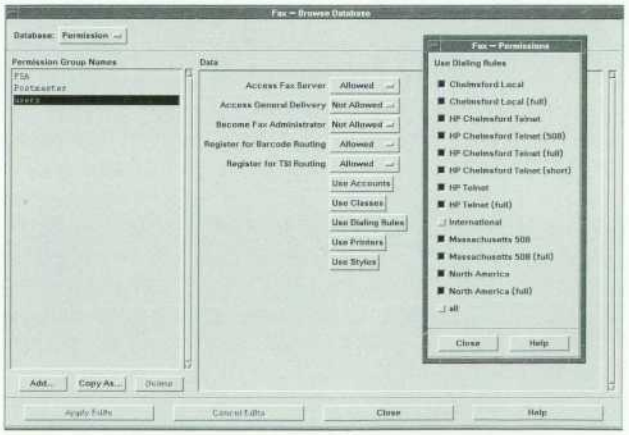


Fig. 4. The major components of the HP MPower fax server architecture.

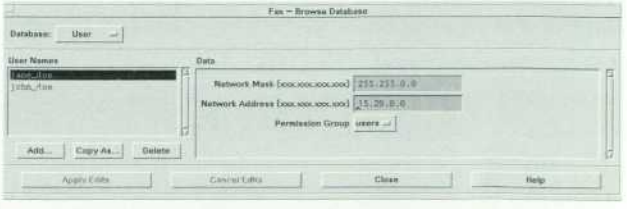
such as least-cost scheduling and incoming fax routing. These databases can be grouped into three major groups: user databases, scheduling databases, and routing databases.

User Databases. The databases used by the fax server to maintain access control information about HP MPower fax users include a permission database and a user database (see Fig. 5).

- Permission database. The permission database maintains a list of permission groups into which users can be placed. Capabilities such as access to general delivery and the ability to become a fax administrator can be given to specific permission groups. Permissions to connect to the fax server, to use specific accounts, classes, dialing rules, printers, and styles, and to register for a bar code or a caller identifier are all granted on a group basis. (Bar codes and caller identifiers are used for routing incoming faxes.)
- User database. To connect successfully from a fax client to the fax server, the user must be listed in the user database and belong to a permission group that has permission to connect to the fax server. The user database also controls the range of network addresses from which a user can initiate a connection request. With each user entry, a network mask and a network address are kept. When a user invokes the fax client, a connection request to the fax server is initiated. If the result of ANDing the network address of the user's workstation with the network mask listed in the database entry for that user matches the network address in that entry, the connection is accepted. Otherwise, the connection is rejected.



(a)



(b)

Fig. 5. Fax user databases. (a) Entries in a permission database. (b) Entries in a user database.

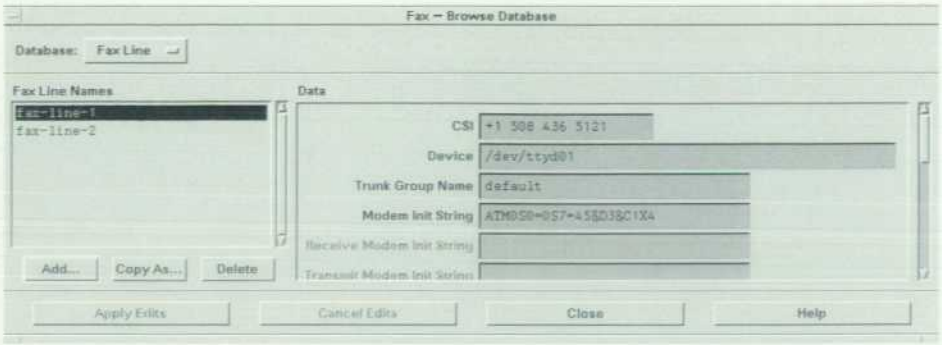


Fig. 6. Typical entries in a fax-line database.

Scheduling Databases. The following databases are used by the fax server when scheduling outgoing faxes:

- Account database. Each outgoing fax submitted by any fax client is required to specify an account to which the call will be charged. Each outgoing fax is logged in the account log together with the duration of the call and the account to be charged. The account database maintains the list of accounts available to users.
- Class database. One of the advanced features of the fax server is the ability to schedule an outgoing fax for transmission when the cost of the call will be least expensive. This feature is called least-cost scheduling, which is controlled by the specification of the class of service defined for the outgoing fax. The class of service also controls which phone lines, if there are more than one in the system, are allowed to be used for transmission. If a fax transmission should fail because of a busy signal or no answer at the receiving end, retry attempts will be made later. The class of service also controls the time between retries as well as the maximum number of retries permissible. The class database maintains the various features provided for each class of service.
- Style database. The style specified with each outgoing fax controls the resolution used for the transmission, the paper length, and the appearance of the fax cover sheet. Such information is maintained in the style database.
- Fax-line database. The fax-line database tells the fax server how many fax modems have been configured for use by the fax server. It also contains information necessary for initializing the fax modems. If the system has more than one fax line, not all of them behave identically. For example, some fax lines might be connected to regular telephone lines and some to leased lines. Because of the different transmission lines, the fax-line database contains entries called trunk groups, which join together fax lines that have identical behavior. When an outgoing fax is eligible for transmission, the fax scheduler will pick the unused and least expensive fax line from among the trunk groups allowed by the particular class of service. Fig. 6 shows the entries in a typical fax-line database.
- Tariff database. Local telephone companies and long distance carriers charge different rates depending on the time the call is made and whether the call is local or long distance (see Fig. 7). The tariff database is used for maintaining such information about these rates. This information is required for the least-cost scheduling feature described above for the class database.

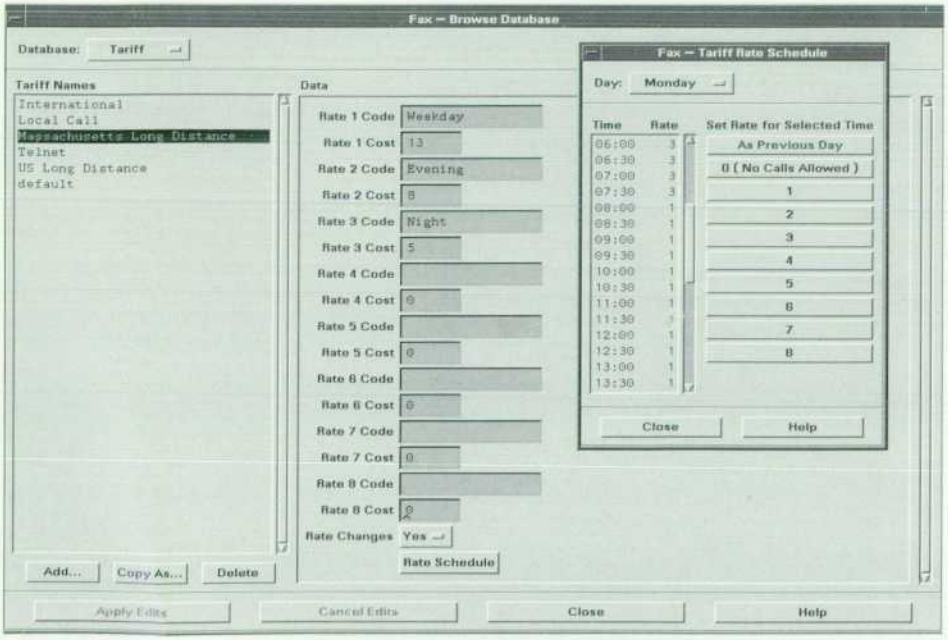


Fig. 7. Entries in a tariff database.

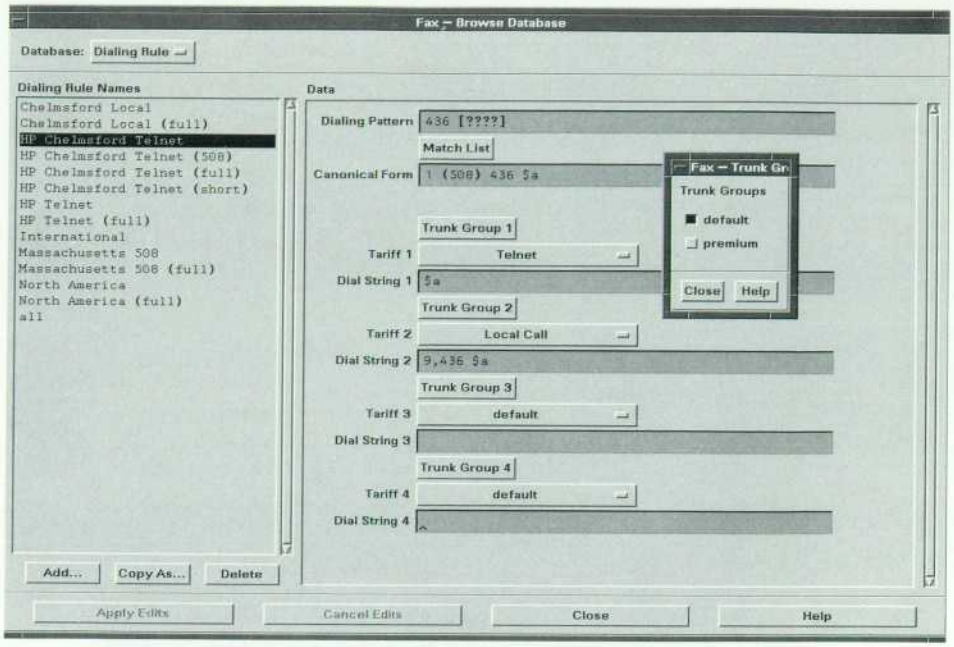


Fig. 8. Entries in a dialing-rule database.

- Dialing-rule database. Another advanced feature of the fax server is the ability to convert any given fax number into the appropriate string of numbers to be dialed. With a properly set up dialing-rule database, the fax server can determine if it can ignore the area code in a 10-digit fax number if the number represents a local call. The server can also determine if a fax number needs extra numbers to add a long distance access code. With pattern matching capability, the dialing-rule database has the flexibility to fulfill the dial string conversion needs of very different systems such as internal telnet systems that require outside line access codes, or systems that have different access codes for various long distance carriers. Another function of the dialing-rule database is to associate each converted dial string with a particular trunk group and tariff entry. Through this association, the fax scheduler can determine the least-costly way to send a fax. Fig. 8 shows the entries in a typical dialing-rule database.
- Printer database. The printer database maintains the list of printers available to the fax server and the associated commands required to print the acknowledgment of an outgoing fax. When a fax is submitted, the user can select to print an acknowledgment of the fax.

The fax scheduling databases are summarized in Fig. 9.

Routing Databases. The routing databases are used by the fax server to control how incoming faxes are handled for each user.

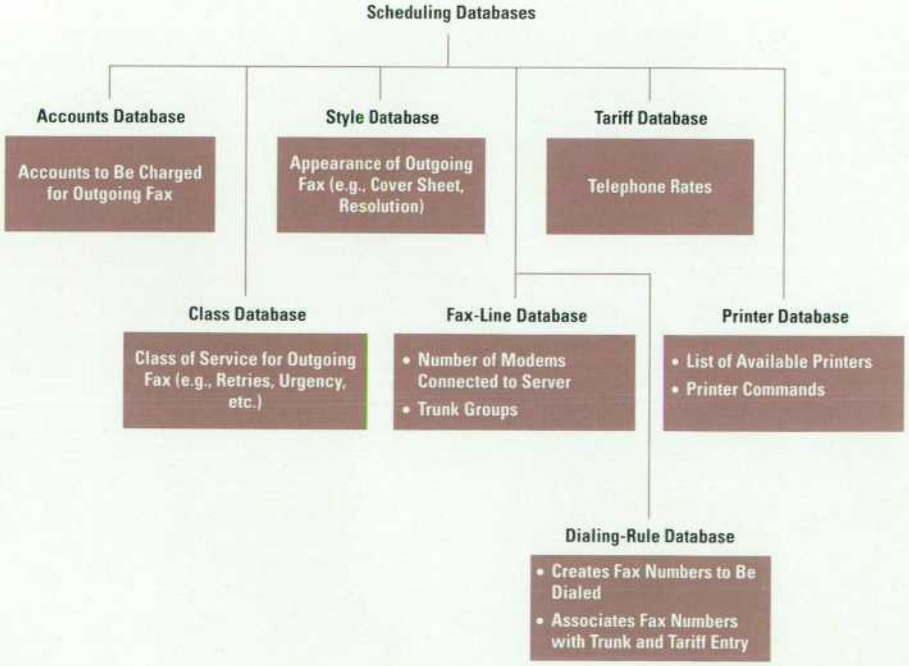


Fig. 9. The relationships between the scheduling databases.

- Action database. The action database maintains actions to be performed on incoming faxes on a per-user basis. Each user can specify a number of action entries, all of which are private to the user. Each action entry may include any or all of the following types of actions:
  - Queue the fax to the in-box of a particular user
  - Send email to an email address
  - Forward the received fax to a fax number
  - Execute some shell command.

  The most commonly used action is queueing the received fax to the in-box of the user who owns the action entry.
- Dispatch database. The dispatch database is presented to the user as bar-code registration and caller-identifier registration. Bar codes are typically associated with faxes sent from workstations and caller identifiers are typically the telephone numbers associated with standalone fax machines. By registering a bar code or a caller identifier, the user defines an action entry to be performed by the fax server when an incoming fax has a bar code on the cover sheet or a caller identifier from a fax machine that is similar to what is being registered. The user may choose to allow other users to register for the same bar code or caller identifier if it has not been registered and disallowed by another user. Typically, bar codes are not shared, while caller identifiers may be shared.

Default Configuration. In the default configuration, the user database is empty. The fax server will accept any connection requests from any fax clients originating from anywhere on the network. Any user can submit outgoing fax requests using any of the accounts, classes, and styles present at the fax server. The default dialing-rule database has one entry that will match any fax number and dial it as is. The default action and dispatch databases are also empty. Thus, all incoming faxes will be left in the general-delivery area. Any user is allowed to access the general-delivery area to check for received faxes. In this default configuration, the behavior of the fax server is very much like that of a standalone fax machine.

Fax Server Components

The fax server components shown in Fig. 4 contain the processes that interact with the fax databases to handle communication with fax clients, the transmission of faxes, and the reception and distribution of incoming faxes.

Client-Connection Component. The client-connection component is the process that communicates directly with fax clients. It accepts or rejects connection requests from fax clients according to information stored in the user and permission databases. Once the connection has been established, the user at the fax client can submit outgoing fax requests as well as look at received faxes placed in the client's in-box folder. With the right permission, the user can also check and claim received faxes stored in the general-delivery folder. If the user is a fax administrator, a received fax can be routed from the general-delivery folder to the in-box folder of another user.

When an outgoing fax request is submitted to the client-connection component, the permission database is accessed to see if the user is authorized to use the account, class, and style specified in the request. The dialing-rule database is accessed to see if the fax number matches any of the dialing rules the user is allowed to use. From the style specified, the cover sheet is located for the request. After these checks the appropriate fax renderers are invoked to render the cover sheet and attachments (if any) into TIFF Class F images. These images and a command file are kept in the fax spooling area under the directory /usr/spool/fax/destinations. The transmission component described below will scan this area and schedule the fax for transmission. Fax renderers provided with HP MPower support ASCII, PCL, and PostScript files, and all EFS file types supported by the HP Image Library, including TIFF, GIF, JFIF, JPEG, Xpm, Xwd, and Xbm images. The HP Image Library is described in the article on page 37.

In addition to submitting and retrieving faxes, other administrative operations such as access or modification to any entries of the databases are also executed over the connection between the fax client and the client-connection component. The user needs to be a fax administrator to request such operations.

The establishment of a connection between the fax client and the client-connection component is mediated by the internet daemon inetd. Appropriate entries are made in the files /etc/services and /etc/inetd.conf when HP MPower is installed to let inetd know how the connection can be established. If the NIS† (network information service) is used, the NIS master versions of these two files must be modified to include these entries.

Transmission Component. The transmission component consists of the processes that are responsible for scheduling outgoing faxes and transmitting the rendered images to the fax modem. The fax scheduler process is responsible for scheduling, while the fax renderer and fax transmitter processes are responsible for rendering and transmitting.

The fax scheduler is run as a background process. Only one such process should run on the system on which the fax server is running. It wakes up every minute and scans the fax spooling area where all the submitted and rendered faxes are kept. The command file of each outgoing fax contains the class of service information (defined in the class database) that the fax scheduler uses to determine whether a particular fax is eligible for immediate transmission or not. If the fax is eligible for immediate transmission, the fax scheduler checks to see if any of the fax lines that the user is authorized to use are free. If a free line is found, the fax transmitter is invoked to start the fax transmission on that line.

If the fax transmission fails because the destination fax number is busy or does not answer, the fax will not be eligible for retransmission until the Busy_Retry_Interval or No_Answer_Retry_Interval specified by the class of service for the transmitted fax elapses. If the fax transmission fails because of transmission error, the fax will not be eligible for retransmission until the Tx_Failed_Retry_Interval elapses. Fax transmissions that fail after the successful transmission of a number of pages will continue from the failed page when retrying transmission. If the fax transmission failure reaches the respective retry limit specified by the class of service, the fax will be put in the suspended queue. The user then has the option to send it again to the same fax number or to try a different fax number.

† The network information service provides global administration of large network systems.

Both successful and failed fax transmissions are logged in the account log together with the duration of the calls and the accounts used. This helps to rectify fax phone charges to specific cost centers. When accessing the account log, regular users will see only the fax transmissions they submitted, while fax administrators will see all the entries.

Other operation status information is also logged by the fax scheduler and fax transmitter in the system log for diagnostic purposes. Only fax administrators can access the system log.

Reception Component. The reception component consists of the processes responsible for monitoring incoming fax calls, receiving incoming fax images, and dispatching the received faxes. These tasks are accomplished by the fax listener, the fax receiver, and the fax dispatcher processes respectively.

The fax listener is run as a background process. Each fax line on the system is monitored by an individual fax listener. If an incoming call is detected, the fax receiver is invoked to receive the fax images on that line. The fax receiver and the fax transmitter are located in the same program.

When reception of an incoming fax is finished, the fax dispatcher is invoked to determine how to dispose of the newly received fax. The fax dispatcher provides one of the advanced features of the fax server called *incoming fax routing* in which an incoming fax will be automatically routed to its recipient if there is routing information available with the fax. This feature is described in detail later. If no routing information is available, the received fax is placed in the general-delivery folder. Users with general-delivery access permissions are able to view the first page of received faxes in the general-delivery folder and claim them or route them to the correct recipients. This mimics the behavior of a standalone fax machine.

Incoming faxes are logged in the account log. Only fax administrators can see the entries for incoming faxes in the account log.

HP MPower Fax Advanced Features

The advanced features currently provided in HP MPower fax include the ability to pull together several pieces of information to determine the most cost-effective way to schedule and transmit outgoing faxes and provide automatic routing of incoming faxes.

Least-Cost Scheduling. When an outgoing fax is submitted, the fax server checks the class specified for the fax to determine which trunk groups can be used for transmission. Next, the fax number is checked against entries in the dialing-rule database that the user has permission to use. The dialing-rule entry that matches most of the digits in the fax number (not counting those matched with patterns) is used if it can be used in at least one of the trunk groups found above. For each trunk group specified in the dialing-rule entry, the associated tariff is noted and used for determining the least expensive way to transmit the fax. The fax server then makes a table of time periods within which the fax is allowed to be transmitted, together with the costs associated with each time period. This information is stored in the command file for that fax.

When the fax scheduler checks the command file of each outgoing fax to see if the fax is eligible for immediate transmission, it first checks to see if the user selected the Wait_to_Send option. If this option is used and the specified time has not arrived yet, the fax is not eligible for immediate transmission. One reason for choosing the Wait_to_Send option might be to avoid transmitting when the the receiving fax machine is busy. A fax can be delayed for up to 24 hours.

If the specified time arrives for the Wait_to_Send option or that option was not chosen in the first place, then the deadline for transmitting that fax is checked. The deadline for transmission is specified in the class for the fax. It is measured from the time specified in the Wait_to_Send option, or from the time when the fax is submitted if the Wait_to_Send option is not chosen. If the deadline for transmission passes, the fax scheduler will schedule the fax for immediate transmission, ignoring cost. If the transmission deadline has not passed, the fax scheduler will consult the table of time periods in the command file to see if the current time falls in the least-expensive time period before the deadline for transmission arrives. If the current time is in the least-expensive range, the fax is eligible for immediate transmission.

Incoming Fax Routing. When an incoming fax has been received, the first page, usually the cover sheet, of the received fax is scanned by the fax dispatcher for a bar code. If one is found, the dispatch database is checked to see if such a bar code has been registered by one or more users. If found, the actions associated with each bar-code registration will be performed. Next, the dispatch database is checked to see if the caller identifier of the received fax is registered to one or more users. If found, the actions associated with each caller-identifier registration will be performed. Such actions may include any combination of routing the received fax to the in-box folder of the user, sending email notification to the user, forwarding the received fax to a different fax number, or executing a user-specified shell script. Forwarding a received fax to a different fax number is very useful if the fax recipient is temporarily at a different site that can receive fax.

With incoming fax routing, faxes that are directed to a particular user, say by a bar code, can be placed in the user's in-box folder immediately without the delays usually involved when it is placed in the general-delivery area waiting for someone to dispatch it manually. Avoiding the general-delivery area also provides better confidentiality.

Other Interfaces to HP MPower Fax Server

Apart from sending and receiving faxes via the fax client, two other interfaces are available for submitting outgoing faxes. One of them is the fax email daemon, and the other is the simulated printer interface. Note that these interfaces provide only a small subset of the functionality and features of the fax client.

Fax Email Daemon. The fax email daemon is run as a background process on the machine on which the fax server is running. Users who are on a system that cannot execute the fax client can send email to faxemd@<fax_server>, where <fax_server> is the machine on which the fax server, and thus

the fax email daemon, is running. By providing the fax-specific header information in the beginning of the email, the body of the email will be sent as a fax by the fax email daemon. The following keyword-argument pairs are required in the fax-specific header:

Keyword	Argument
From:	Sender name, sender title
Style:	Style name
Class:	Class name
Account:	Account name
To:	Recipient name, recipient title
Fax#:	Fax number

Only the ASCII contents of the body of the email can be sent. File attachments are not supported.

Simulated Printer Interface

The simulated printer interface, faxlp, allows users to submit outgoing faxes as though they are queueing to a printer. The command to use is:

```
lp –dfaxlp –o'from: Jane Doe' –o'class: Standard' –o... the_file
```

where the_file is the file that will be sent as a fax. Here, the_file may be an ASCII file or a PCL file. This interface allows the addition of fax capability to applications that support a printer interface without having to make modifications to the applications to accommodate a fax machine.

Conclusion

The client/server architecture of the HP MPower fax product provides an easy-to-use but full-featured fax capability to HP 9000 computers. Its advanced feature of least-cost scheduling can save on telephone costs. By minimizing the need for human attention, productivity of everyone is increased. Finally, HP MPower fax enables users to use their fax resources more cost-effectively.

Audio Support in HP MPower

Multimedia capability promises to enhance the communication and presentation of information through the use of real-world data types such as audio and video. Compact-disk-quality audio is the first of such data types to be offered as a standard feature on all of HP's new workstations.

by Ellen N. Brandt, Thomas G. Fincher, and Monish S. Shah

HP MPower provides the hardware and software to allow recording and playing of audio files over a network, incorporating audio in email, adding audio annotations to system files, and recording and playing to external devices like tape recorders, CD players, and VCRs. With these capabilities audio-enabled applications can add voice annotation to documents ranging from spreadsheet rows and columns to CAD drawings. Programmers might add audio comments to their programs. Error messages could take the form of spoken messages, or even distinctive sounds that convey more information than a simple beep. Finally, background music could be added to presentations.

This article describes HP MPower's audio functionality, application development tools, and audio hardware and software architecture.

Audio Tools

The audio tools provided in HP MPower allow users to record, edit, and play audio data in a variety of file and data formats. HP MPower also provides tools for converting between audio formats. All of these tools are built on the audio library, which defines the application program interface to HP MPower's client/server audio implementation. The application program interface, libraries, widgets, and header files are available to third-party software developers who wish to use audio in their applications.

The Audio Editor. The audio editor is based on OSF/Motif widgets and audio library (alib) functions. The audio editor enables the user to record, play, and edit audio files in a variety of file and data formats. It displays a waveform representation of the data to make editing easy and supports basic editing tasks like selecting, cutting, and pasting. The main screen for the audio editor is shown in Fig. 1a.

The audio editor can be invoked or redisplayed by clicking on the audio icon on the HP MPower media bar. Dropping a file from the HP VUE file manager onto the audio icon will bring up the audio editor with the file already loaded and

(a)

(b)

Fig. 1. The audio user interface in HP MPower. (a) The audio editor. (b) The audio control panel.

displayed. A file can be loaded into an already visible audio editor by dropping the file icon onto the editor screen.

Audio Control Panel. The audio control panel is an OSF/Motif interface to global audio parameters such as volume and output device selection (see Fig. 1b). All cooperating audio applications can use the control panel so the user always knows where to go to control these attributes. The audio control panel also provides a Stop button that will stop the current play operation from any cooperating application.

The audio control panel allows the user to turn monitoring on or off. Monitoring involves listening to the audio input signal. In a conventional tape recorder, monitoring allows the user to listen to the audio signal being recorded. On a workstation that has HP MPower, monitoring can be used whether or not anything is being recorded. For example, a user can monitor the workstation's line inputs from a CD player or VCR even when recording is not in progress without using the CPU or other system resources.

Audio Playback. An audio file can be played by simply double-clicking on the file's icon in the file manager or in an audio-enriched mail message. The file will begin playing according to the settings of the audio control panel. A file can also be played by dragging its icon from the file manager to the speaker icon on the HP VUE front panel.

Other Functionality. In addition to the graphical interfaces to audio functionality mentioned above, there are some other capabilities shipped with HP MPower that are accessible from a command line. In the directory /usr/audio/bin executables for the audio editor (audio_editor), the audio control panel (AudioCP), the double-click function (send_sound), and the convert and attributes programs are provided. The convert program converts audio files from any supported file format, data format, and rate and number of channels to any other format and rate. The attributes program tells everything that it can determine or guess about any audio file including file format, data format, sampling rate, number of channels, data length, and header length.

Audio Data and File Types

A number of audio data and file types exist in the industry today, making it difficult for audio files to be shared in a heterogeneous environment.

The lack of standards for audio is currently being addressed by groups such as the IMA (Interactive Multimedia Association). This organization and others are trying to develop standards so that someday there will be a clearer picture as to how to store audio information in a format that is accessible to everyone and can be easily incorporated with other aspects of multimedia.

In the meantime, we chose to support two of the existing file formats and to develop conversion utilities that allow us to support sampling rates, data formats, and byte ordering methods that are not supported directly by our audio hardware.

File Format. A file format is a structure in which there is information about the data as well as the data itself. This information may reside exclusively in a header at the beginning of the file or may be interspersed in "chunks" throughout the file. In the latter case, it is possible that only certain types of chunks are pertinent to audio.

Pertinent information for audio files includes the sampling rate, data format, number of channels, compression technique, and number of samples. Sometimes there is additional information such as loop points or edit markers.

The two audio file formats we chose to support include Microsoft® RIFF/Waveform, which is chunk-based, and the NeXT/Sun audio format, which is a file header followed by data.

We also support audio files that contain only the raw samples. Although this is a popular method of storing audio data and it can be useful in a heterogeneous environment, we recommend using a file format with a header whenever possible because the attributes of the audio data cannot be determined from the samples themselves.

Data Format. Data format defines the method in which audio samples are stored. The most basic method is linear PCM (pulse code modulation). The signed amplitude of the audio waveform is quantized at fixed intervals of time. Each step of the quantization has equal size. This format is very easy to work with and is preferred by most audio editing and mixing routines. One of the formats that our audio hardware supports is the 16-bit linear PCM, which is used by compact disks and is popular on UNIX*-system-based workstations.

An unsigned version of 8-bit linear PCM is very popular among Macintosh and PC users because instead of having an amplitude range of −128 to +127, unsigned (or offset) 8-bit linear has an amplitude range of 0 to 255. All samples are offset by 128. For example, silence, which is normally quantized as 0, is recorded as 128.

We also support the CCITT (International Consultative Committee for Telephone and Telegraph) μ-law and A-law standards. These are companded PCM formats. In these formats the straightforward linear scale is replaced with a base-eight logarithmic scale such that there are small step sizes at low signal levels and large step sizes at high signal levels. The result is a better signal-to-noise ratio and the ability to represent the dynamic range of 13-bit or 14-bit samples with only 8 bits. μ-law and A-law both specify 8-bit samples at 8000 samples per second. μ-law is very common on UNIX-system-based workstations. An overview of the A-law and μ-law data formats is provided on page 65.

Sample Rate. The sample rate refers to the number of digital samples used to represent one second of analog audio. The greater the number of samples, the more accurately the audio signal will be reproduced. A sample rate of 8 kHz (8000 samples/s) can reproduce human voice with adequate clarity, but it does a very poor job on music. For music, 44.1 kHz (44,100 samples/s) works well. The music on all compact disks is recorded at this sample rate. Digital audio tapes (DAT) have a 48-kHz sample rate, producing slightly better audio quality. The audio hardware (described later) supports all of these sample rates plus a few others.

The sampling rate is also subject to the Nyquist criterion, which dictates that the sampling rate must be at least twice the rate of the highest frequency being recorded. Higher frequencies will typically be filtered out by the recording hardware.

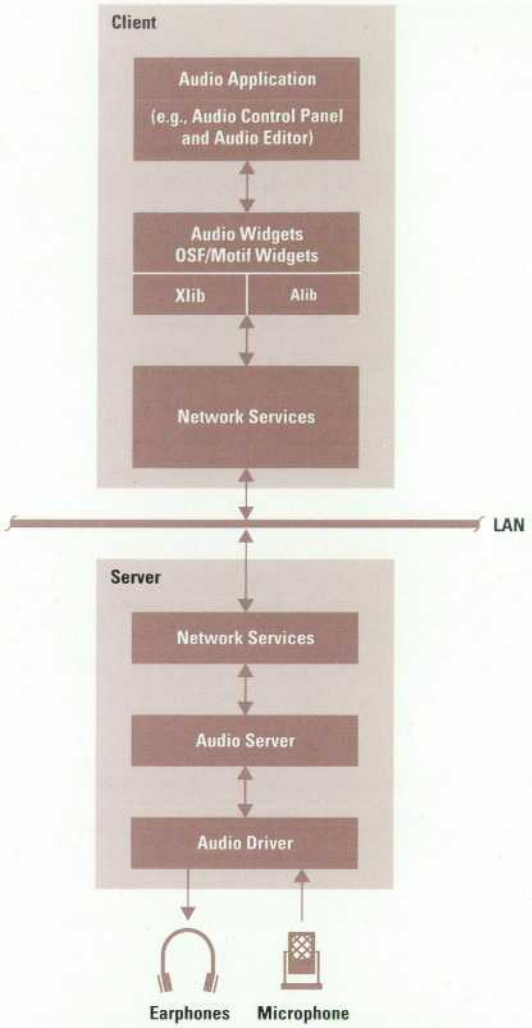


Fig. 2. Client/server architecture for the audio software.

Multiple Channels. Although audio files with more than two channels do exist, most are either mono (one channel) or stereo (two channels). The two channels in a stereo file are typically interleaved on a sample by sample basis. This means there will be a sample for the left channel, followed by one for the right, followed by the next one for the left and so on.

Byte Ordering. One problem with supporting files across a heterogeneous environment is that the byte ordering of the local hardware may be different. In our audio structure we supply information about the byte ordering of the audio hardware connected to the system. Unfortunately, there is no easy way to determine the byte ordering of the audio data in a file that is imported from elsewhere. Therefore we are forced to make assumptions. We assume all RIFF/Waveform files use least-significant-byte-first order and that all other files use most-significant-byte-first order. This applies even to the files created by our audio tools.

Client/Server Architecture

The audio system software uses a client/server architecture that is modeled after the X Window System (see Fig. 2). The client side provides a consistent interface to application programs and handles the connection to the server, which may be local or remote. The server side provides a consistent interface to the client side, handles control of the audio device, and performs data I/O.

The audio server (aserver) interfaces multiple clients to the audio hardware, allowing simultaneous play and record, queuing of multiple play and record transactions, priority preemption, and dynamic buffering.

The audio server also supplies information to the client side about the attributes of the connected audio hardware. The audio library provides the capability to convert various audio attributes to ones that are supported by the connected hardware. Therefore, the hardware attributes (sampling rates, data formats, byte ordering, etc.) do not need to be the same on the client and the server.

Support for Application Developers

Since our audio software subsystem is closely modeled after the X Window System and OSF/Motif, we provide a library, a server, widgets, and a toolkit that are very similar to their X counterparts. When appropriate, we followed the X conventions in our implementation of the various components. For example, the naming convention, function argument convention, and event and error handling of the audio library all follow the conventions used in Xlib. An application developer who is familiar with X and OSF/Motif should find it easy to incorporate audio functionality.

The audio server handles the interface to the audio driver and provides control of audio transactions. This isolates the audio client from hardware-specific device calls and allows a higher level of transaction control than would otherwise be possible.

The audio library is conceptually much like the X library, or Xlib. The audio library provides the low-level application program interface for audio. The audio library provides functions that allow an application to connect to one or more audio servers, to manipulate the configuration of the audio hardware controlled by the servers, to control recording from a server to a file or data stream, and to control playback from a file or data stream to a server. The audio library also provides the capability for applications to play audio from any of several popular file and data formats and to save audio in any of those formats.

The audio widgets provide high-level access to record and play functionality. The application developer can use the widgets without having to learn the lower-level audio library calls. The audio widgets don't export all the functionality of the audio library, but they do provide some flexibility through the use of resources that can be specified by the client application.

The audio toolkit provides callbacks for audio events. Since it is not desirable for audio events to interfere with other events when an application is using the X toolkit, the audio toolkit allows the X toolkit to detect audio events and call the audio toolkit event handler.

Several example programs are also provided for developers. These include the source code and Makefiles. Some of the examples use the widgets and toolkit, and others use only the low-level audio library calls.

(continued on page 66)

Overview of A-law and μ-law Data Formats

An analog audio signal is continuous in time and amplitude (Fig. 1a). In contrast, a digital audio signal is discrete in time and amplitude. An analog-to-digital converter is used to convert an analog audio signal to digital format. This conversion consists of two separate steps: sampling and quantization. Sampling converts the analog signal from continuous time to discrete time by capturing the value of the analog signal at regular intervals in time. Theoretically, the sampled signal only has a value at those sampling times. Fig. 1b shows the sampled version of the signal in Fig. 1a.

The process of quantization maps each sample value to a number. These numbers are represented with a fixed number of bits, giving them limited precision. The quantization process picks the number that best approximates the amplitude of the sample. Essentially, each step in the digital code represents a quantum of increase (or decrease) in the amplitude. Hence the name, quantization. Fig. 1c shows the signal after quantization, along with the digital codes assigned to each quantization level. A digital audio stream is simply a sequence of such digital codes. Although Fig. 1c shows only a few quantization levels, actual implementations use thousands of levels.

The most straightforward form of quantization uses fixed-size quanta of amplitude. In other words, each step in the digital code represents the same step size in amplitude. In this scheme, the mapping from amplitude to digital codes can be represented with a linear function (known as linear quantization). Fig. 2a shows a linear mapping function.

A-law and μ-law define a kind of mapping in which the digital code is roughly equal to the logarithm of the amplitude. Although both laws use the same basic idea, the actual mapping equations are somewhat different in the two laws. Also,

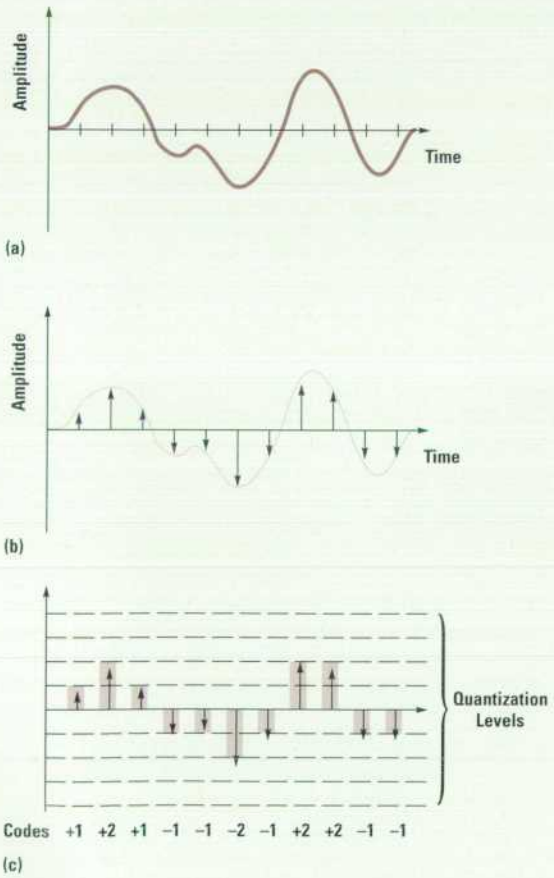


Fig. 1. The stages of converting an audio analog signal from analog to digital format. (a) Original analog signal. (b) Sampled version of the analog signal. (c) Quantization levels assigned to the sample points.

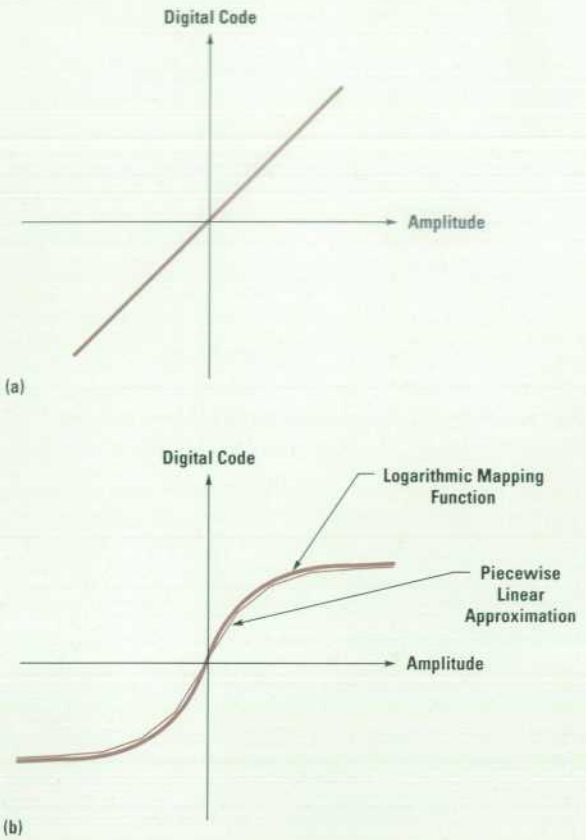


Fig. 2. Methods of quantization. (a) Linear mapping, or linear quantization. (b) Logarithmic mapping function and its piecewise linear approximation.

to simplify the conversion to or from linear quantization, a piecewise linear approximation is used. Fig. 2b shows a logarithmic mapping function and its piecewise linear approximation.

Telephone companies use A-law and μ-law to carry long distance conversations because these formats save bandwidth. For telephone conversations, signal strength may vary as much as 30 dB because some callers have softer voices than others, or some microphones are more sensitive than others. It is necessary to maintain a 35-dB signal-to-noise ratio (SNR) over the entire range. With linear mapping, quantization noise remains independent of the amplitude level, so one must design for 65-dB signal-to-noise ratio (30 dB + 35 dB) to meet the requirements. As the signal strength varies over the 30-dB dynamic range, the SNR varies between 35 dB and 65 dB. This solution delivers a higher SNR than required at most amplitudes to meet the SNR requirement at minimum amplitude. The price for exceeding the SNR specification in a linear mapping scheme is that 12 bits per sample would be required. However, using A-law or μ-law, 35-dB signal-to-noise ratio can be maintained over a 30-dB amplitude range with just eight bits. This works because logarithmic mapping has the property that larger signal amplitudes result in larger quantization steps. Thus, quantization noise is proportional to the amplitude, maintaining a reasonable SNR across a broad dynamic range. This is a more efficient way to use the quantization levels, making eight bits per sample adequate. Thus, the use of A-law or μ-law results in a 33% reduction in telephone audio transmission. The same benefits are realized in computer audio.

The differences between A-law and μ-law are minimal. The U.S.A., Canada, and Japan use μ-law for telephone transmission, whereas most other countries use A-law for telephone transmission. While A-law is somewhat easier to implement, μ-law provides slightly better quality at low amplitudes. Note that A-law and μ-law work well with voice audio only. For high-fidelity music reproduction, linear mapping with 16 bits per sample is typically used.

Audio Hardware

While audio components such as microphones and speakers work with analog signals, the workstation CPU manipulates data in digital form. Thus, to record audio on a workstation, the analog audio signal must be converted to digital form using an analog-to-digital converter. Similarly, audio playback requires a digital-to-analog converter. The addition of these components turns a workstation into a versatile tape recorder—one that can provide rapid access to a large number of audio clips and associate them with other data types within the workstation.

By supporting several options for sample rates and data formats and offering a choice of stereo or monophonic sound, the audio hardware used on HP 9000 workstations allows the user to make the appropriate trade-offs between quality and storage requirements. For example, higher sample rates result in better quality, but also require more storage. The user has the freedom to choose the appropriate quality.

The audio inputs and outputs are compatible with most consumer equipment, which allows easy connectivity. Two types of inputs are offered on our audio hardware: microphone and line in (for VCRs and compact disks). Only one of these may be selected at any given time. Three outputs are available: speaker, headphones, and line out. Any combination of outputs may be activated simultaneously, but they will all output the same signal. Although the audio design supports these five types of inputs and outputs, some workstations do not contain all five connectors because of space restrictions. Input and output may be activated simultaneously, which means that simultaneous recording and playback is allowed.

Audio under the UNIX Operating System

Audio presents a special challenge to a UNIX system because audio is an isochronous data type. Isochronous means constant with respect to time. This implies that the system must process audio samples at exactly the specified sample rate. If it slows down or speeds up, the listener will perceive distortion in the audio. Unfortunately, the UNIX operating system was not designed to work with isochronous data types. In fact, the UNIX system supports multitasking, which means that the CPU splits its effort between any number of tasks that might be active. Obviously, when there are more tasks outstanding, each task gets less time from the processor. This makes it impossible to guarantee that the audio hardware will get as much processor time as it needs.

Even the hardware infrastructure of the workstation can interfere with audio operation. Just as the CPU cannot guarantee adequate attention to audio, the system bus cannot guarantee adequate bandwidth for audio. The audio hardware design team's challenge was clear: overcome these difficulties while maintaining low cost.

The Solution

The problem description above makes it clear that a solution that guarantees isochronous operation under all conditions does not exist. The designers instead chose an approach that achieves isochronous operation under most conditions. That approach calls for putting a reasonable, not worst-case, upper bound on the delay for any given operation and computing how much audio data could be consumed or produced

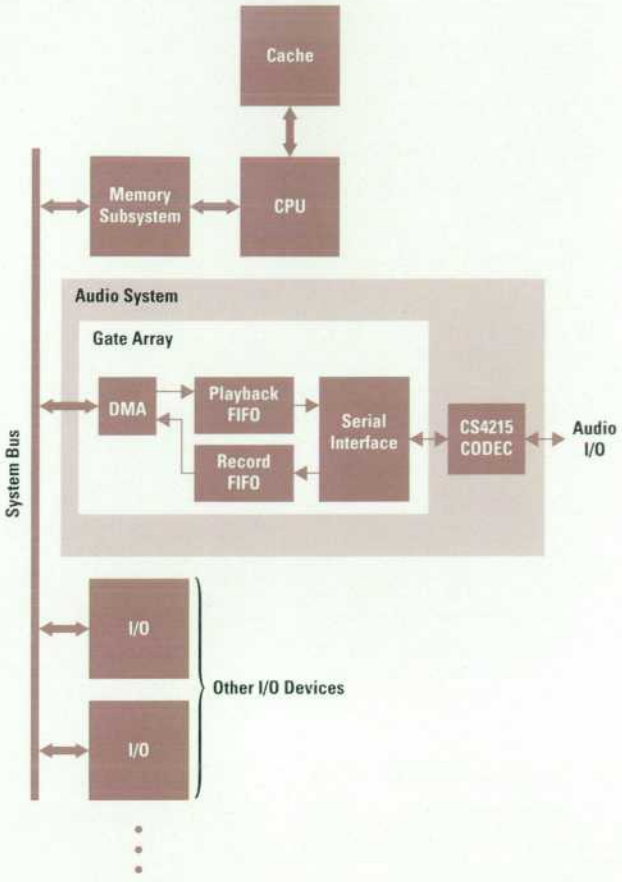


Fig. 3. Block diagram of the audio I/O hardware within a simplified representation of a portion of a workstation.

in that time. A FIFO buffer of that size is required to cover that delay. For example, a heavily loaded CPU may shift its attention away from the audio tasks for one or two seconds. Therefore, more than two seconds of audio is typically buffered in system memory. (Actually, the user can vary the size of that buffer if necessary. A system that is often heavily loaded might need a larger buffer.)

Two options exist for moving audio data between system memory and the audio hardware. Either the CPU can move it in response to an interrupt, or the audio hardware itself can move it. Since the CPU often turns its attention to other things, it might take several milliseconds to respond to an interrupt. Under the design philosophy described above, the audio hardware would need to buffer enough data to cover that delay. To provide adequate storage space, a separate memory chip would be required, making the audio hardware somewhat more expensive. On the other hand, the audio hardware could access the data using direct memory access (DMA). To perform DMA, the audio hardware must become the master of the system bus. The delay for getting mastership is on the order of tens of microseconds, and the buffer must be able to cover that delay. The DMA approach reduces the size of the buffer by two orders of magnitude. In fact, in the final design, the required buffer size is small enough to implement inside a simple gate array (see Fig. 3). The gate array is required to interface to the bus anyway, so the cost increase was negligible. Of course, the DMA logic

added design complexity. Thus, cost was reduced through increased R&D effort. To make the audio hardware inexpensive enough to offer it as a standard feature, the engineering trade-offs had to favor lower cost.

Physical or Virtual DMA?

All modern workstations employ a virtual memory system that allows the CPU to work with virtual memory spaces that are larger than the physical memory available in the system. All programs access memory using a virtual address, and the CPU hardware translates it to a physical address. In contrast, DMA accesses do not go through that same hardware, so the hardware initiating a DMA request must supply the physical address to the bus. In that sense, HP workstations do physical DMA, not virtual DMA. Still, an I/O device could include the hardware necessary to do virtual-to-physical translation, effectively giving that device the ability to do virtual DMA. Since programs work with virtual addresses, they could communicate with I/O hardware more easily if that hardware did virtual, rather than physical DMA.

Unfortunately, virtual-to-physical translation hardware adds complexity and cost to I/O hardware. For that reason, our audio hardware does not do virtual DMA. Instead, the driver software assumes the responsibility of presenting the hardware with physical addresses. Again, the trade-off was made in favor of lower system cost.

Hardware Components

Fig. 3 shows a block diagram of the audio hardware components within a simplified representation of a workstation. As shown in the figure, the audio hardware connects to the workstation's system bus. The CPU uses the system bus to initialize the audio hardware with the desired parameters such as sample rate, volume level, and so on. The DMA block uses the system bus to read audio data for playback and to write audio data for recording. It writes the playback data into the playback FIFO and reads the recorded data from the record FIFO. Each FIFO's size is 8 words by 32 bits. The other ends of the FIFOs connect to the serial interface block. This hardware converts audio data from parallel form, which the FIFOs use, to serial form, which the audio CODEC requires. The term CODEC is an abbreviation for coder/decoder. In this context, analog-to-digital converters are called coders and digital-to-analog converters are called decoders. The audio CODEC implements two converters of each type in a single chip, allowing stereo operation. Some of the CODEC inputs and outputs are buffered with analog amplifiers. In some cases, the amplifiers provide more gain, while in others they help match input or output impedance. The CODEC also implements the logic required to support the various sample rates and data formats discussed earlier. The analog-to-digital and digital-to-analog converters inherently operate with 16-bit linear data. So, if A-law or μ-law mode is selected, the CODEC converts playback data from the selected mode to 16-bit linear and recorded data from 16-bit linear to the selected mode.

The CODEC is a commercially available part. A single gate array implements the rest of the logic in the audio hardware. Some salient features of this HP-designed gate array are:

- 4,953 gates
- 1.0-micrometer technology
- 120 PQFP (plastic quad flat pack) package.

The process used for this gate array allows finer-pitch I/O pads than most similar processes. The finer pitch is better matched to the particular ratio of gates to pins of this chip. If manufactured in another process, this chip would have cost more because of a larger die area.

Conclusion

For audio to be useful on the desktop, audio capabilities must be pervasive. The HP 9000 Series 700 workstations have kept the cost of audio hardware and application development low by avoiding special-purpose hardware like digital signal processors in favor of using the power of the PA-RISC processor to handle digital audio signals. This allows HP to ship high-quality audio with every workstation. Our audio offering is completed with an audio server and basic audio tools to get users started with audio, and a library containing audio widgets, a toolkit, and program examples for application developers.

Bibliography

1. John Bellamy, *Digital Telephony, Second Edition*, John Wiley and Sons, 1991, pp. 110-119.

Video Support in a Multimedia Environment

Combining video with the computing power of a workstation adds an extra level of interpretation, detail, and perception to information seen and manipulated on a workstation desktop.

by Craig S. Richard

We have all heard the expression "a picture is worth a thousand words." Images convey meaning that is difficult to express just using words. For example, consider the difficulty in trying to describe in words a person's looks, a shade of color, a complex object, or a CAT-scan image. You can use a lot of words to create a mental image of the object being described and hope that the words are interpreted as you intended, or you can show a picture. Video takes the value of images a step further by presenting 30 interrelated pictures every second.

We live in a time when video is one of our primary sources of information. We depend on video on a daily basis to provide us with the information we need or desire. The television set is the focal point of many homes. We watch television to see the latest breaking news, to see what the weather is going to be like the next day, to see places and things that we may never be able to see in person, and simply for entertainment. Video gives us the perception of "being there" as we watch it. We experience the event rather than create our own interpretation of it, and we remember the experience in more detail.

Why Video on a Workstation

Obviously, buying an expensive workstation just to watch television reruns, sports events, or any other broadcast television shows doesn't make a lot of sense. So why would it be useful to have video on a workstation? A simple answer is that video can add a new dimension for doing many kinds of work. For example, consider a mechanical test engineer who must validate a computer simulation based on the design data with the actual physical behavior of a mechanical part. Typically, this is done by first analyzing the behavior of the mechanical part on the computer using a 3D modeling package. By having the physical behavior available as video information on the workstation, the mechanical part can be simultaneously analyzed with its computer model. The accuracy of the analysis and the ease and efficiency of testing would be greatly enhanced. This is just one example of the advantage of having video functionality integrated into a workstation, and the added value that computers can bring to video.

Using a computer is inherently an interactive experience. The user provides input through a keyboard, a mouse, or some other input device and the computer responds, takes some action, shows the results of that action, and waits for the user to provide more input. Watching television is a passive experience. The viewer typically doesn't interact with the picture on the screen.

As computer technology and television technology converge, the result is the power and impact of video with the control and interactive capability of computers. This is where the advantages of video capability on workstations become apparent. By combining text, graphics, audio, and video into an interactive presentation, the user can quickly and efficiently gather information on demand with a high rate of retention. This is invaluable for on-the-job training where an employee may need to learn (or relearn) a very specific task very quickly. As an example, take an automobile assembler who may work on engine assembly for a few months and then work on front-end assembly for a few months, and so on. This is a situation in which interactive video and workstation capabilities can be combined with a computer-based training course to provide some quick and inexpensive training as the assembler moves from one type of assembly station to another.

Challenges in Video Technology

Although video provides a vast amount of information, the delivery of video also requires a vast amount of digital data. A video image on a monitor is composed of hundreds of thousands of dots of phosphor (pixels) that are illuminated (at different intensities) by an electron gun as it scans across the monitor. The gun scans horizontally across a line of pixels, and then skips down to the next line and scans horizontally across the new line until the entire monitor is scanned. The electron gun then jumps back to the top of the display and begins scanning across the lines again. The whole process is repeated 30 to 75 times a second depending on the refresh rate of the monitor.

For television, the intensity of each pixel is determined by a voltage representing an analog signal received from a video source such as a VCR or cable TV tuner. The signal changes continuously to represent the color and intensity of each pixel.

In the computer, the intensity of each pixel is determined by a voltage representing a numerical value in a specialized computer memory called a frame buffer. In a simplified black and white model, there would be one bit of memory representing the intensity of each pixel. If the bit is on, the

pixel on the monitor would be white, and if the bit is off, the pixel on the monitor would be black. More information is required to represent color images. To display 256 colors simultaneously (the most common workstation graphics capability), eight bits of information is required for each pixel. To represent true color (four million colors), 24 bits of information is required for each pixel.

To change the color or intensity of a pixel, a different value must be written into the frame buffer memory location that represents that pixel. For static, computer-generated images such as text and graphics, the memory values do not need to be updated frequently. The border around the window, or the icon on the top of the screen may not change for hours. However, for animated sequences, the memory values representing the pixels need to change for each new image. For video, this means that the memory value for each of the hundreds of thousands of pixels has to be changed 30 times a second! In the United States, there is a standard video timing format, NTSC (National Television Standards Committee), which allows for a video image that is 640 pixels wide and 480 pixels high. If the NTSC signal were represented in true color, there would have to be three bytes of information for each pixel. This corresponds to 900K bytes of information for each image, and with images changing 30 times per second, over 26M bytes of information would need to be written every second. This is an enormous amount of information to move around every second.

HP VideoLive Requirements

Video technology in HP MPower has been provided by HP's VideoLive product. VideoLive was developed by RasterOps Corporation exclusively for HP. HP engineers specified the requirements for the video capability, and RasterOps designed and produced a product that met the requirements. There were four main design requirements. The first requirement was to provide full-motion (30 frames per second) video in a window on an HP 9000 Series 700 Workstation monitor. The window had to be movable to any position on the display, uniformly scalable, and occludable by other windows on the display.

The second requirement was that the video product had to work with existing HP graphics subsystems. At the time, most video implementations required a frame buffer that was shared by the graphics and the video. HP produces some of the highest-performance graphics frame buffers in the world, and nobody would be willing to sacrifice high-performance graphics for video functionality.

The third requirement was that displaying video images should not adversely affect system or graphics performance. If the CPU were required to display video information, the system would need to allocate most (if not all) of its processing power to the video. In a shared frame buffer implementation, even though the CPU is not rendering the video, there is contention between the CPU and the video when the frame buffer memory is accessed, resulting in the degradation of graphics performance.

The fourth requirement was that although watching video on a workstation has definite value, the ability to capture the video information has even more value. Therefore, it had to be possible to capture individual video frames in digital form.

VideoLive Hardware

To meet the design requirements, an analog mixing scheme is used to generate video frames on the display. VideoLive is a single slot EISA card. The card has an on-board 1024-by-512-by-24-bit video frame buffer into which an analog video signal is digitized in real time. The digitization is accomplished using a Philips SAA7191 video decoder and an SAA7192 color space converter. The contents of the frame buffer are stored in RGB format (8 bits of red, 8 bits of blue, and 8 bits of green). A Brooktree Bt463 digital-to-analog converter (RAMDAC) is used to generate the analog signal that drives the monitor. Fig. 1 shows the architecture for the VideoLive card.

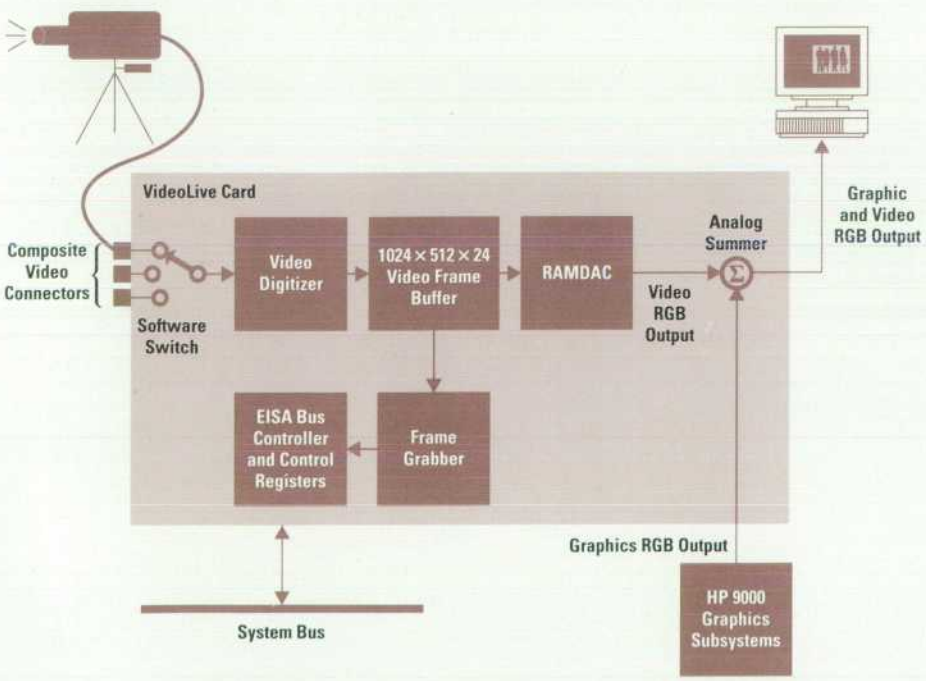


Fig. 1. Simplified block diagram of HP VideoLive hardware.

The RGB output from the graphics frame buffer's RAMDAC is connected directly to the VideoLive card instead of to the monitor. Using a high-speed multiplexer, the RGB output from VideoLive's video frame buffer is mixed with the RGB output from the graphics frame buffer. A set of programmable registers are used to specify the position and size of the video window.

The need to capture digitized video frames is met by digitizing the analog video in real time into a frame buffer. When grabbing frames, the video image is momentarily frozen in the video frame buffer and then the CPU reads the contents of the frame buffer (in RGB format) over the EISA bus and into system memory.

By taking advantage of the HP Image Library capabilities, the captured frames can be saved in a TIFF file and can optionally be compressed using the JPEG compression algorithm. Once in TIFF format, the captured frames can be printed, faxed, and viewed by HP ImageView, and pulled into the HP SharedX Whiteboard application to be viewed and annotated over the network by several people. HP SharedX and Whiteboard are described in the article on page 23 and the HP Image Library is described in the article on page 37.

X Video Software

Live video functionality is tightly integrated into the HP VUE environment. This is accomplished using the X video extensions (Xv) to the X window server (see Fig. 2). Xv is a de facto standard for providing live video functionality for applications based on the the X Window System.

Xv provides the hooks through which X window geometry events such as position changes and clip rectangles can be relayed to the video window. When an Xv window is active, all geometry events to that window are intercepted by the Xv software. The Xv software renders a black area on the screen onto which the video is overlayed. Xv also programs the registers on the video board to position, size, and clip the video.

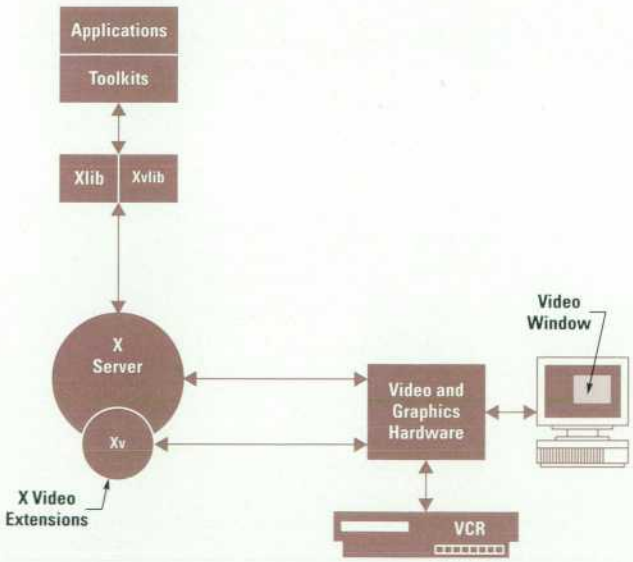


Fig. 2. Client/server architecture and interface points to the video hardware and the workstation display.

Xv also provides a programming interface that allows an application to control the video. The interface provides basic control of turning the video on and off, freezing and capturing video frames, selecting active video connections, and controlling video attributes such as brightness, contrast, hue, and saturation.

Collaboration Using Video Images

As previously mentioned, captured frames of video can be used by any of the image-capable components of HP MPower. This functionality provides powerful collaboration capabilities. How many times have you been on the phone trying to describe a physical object to someone and wished that they could see the object? For example, consider a technician working with a prototype printed circuit board. As frequently happens during early hardware development, a few wires are put on the board to fix layout problems. Suppose the technician is having some problems with the board. The technician contacts the design engineer and describes the problem. The engineer says the problem sounds like a problem that was fixed in an earlier release of the board. The technician and the design engineer can try to fix the problem over the telephone, and if that doesn't solve the problem, either the board or the design engineer must make a trip to fix the problem.

If both the technician and the design engineer are equipped with workstations running HP MPower and live video capability, the technician can point a camera at the defective board or a specific area of the board and capture a frame and save it to a TIFF file. The TIFF file can then be dropped into the HP SharedX Whiteboard and the engineer and technician can share the Whiteboard image. The engineer circles the areas where changes were made and shows the technician how to make the changes.

This is a specific example which can be extended to any situation in which someone is trying to convey information about a physical object.

Conclusion

As computer technology and computational power continue to progress, the capability of processing video information becomes more feasible and less expensive. Video boards are now available from third parties that provide similar functionality to the VideoLive board, with the additional capability of capturing the video data in real time (30 frames per second) and saving the digitized video on disk. HP has released a software digital video player that plays MPEG encoded video files from disk at up to 30 frames per second without additional hardware (see "Digital Video in HP MPower," on page 8).

Mail Facilities in a Multimedia Environment

Providing a multimedia email facility required that the well-established processes of creating, sending, receiving, printing, and replying to email messages be maintained and applied to messages containing multimedia objects.

by Robert B. Williams, Harry K. Phinney, and Kenneth L. Steege

The advent of tools and capabilities that allow users to manipulate and create multimedia objects on a workstation mandated the need to make it possible to send these objects through electronic mail, or email. The user interface for creating, reading, and sending text messages through email is well-established. For multimedia email to be effective the same sort of process flow must be in place. For example, just as a user can use the more command to view a standard text email message, an equivalent facility must be available to view a multimedia email message. What this implies is that the user should not have to be concerned with invoking the correct software to deal with a particular media type because this should be handled by the mail facility.

The HP MPower mail facility, which is represented by the envelope icon on the HP MPower front panel, provides support for sending, replying, viewing, and printing of multimedia mail.

For sending multimedia email, HP MPower provides two approaches: dragging and dropping a file on the envelope icon or clicking on the envelope icon. With the drag-and-drop method, the user is presented with the dialog box shown in Fig. 1a. From this box the user can either send the dropped file to its destination by selecting the Send button or edit the file by selecting the Edit button. If the Edit button is selected, the mail composer (editor) window is displayed showing the contents of the file that was dropped on the mail icon (see Fig. 1b).

To read mail the user can click on the mail icon and be presented with the two windows shown in Fig. 2. The first window contains the standard elm screen and the other contains the HP MPower viewer screen. Elm, which is the HP-UX* screen-oriented electronic mail processing system, provides the user interface for the user to interact with the HP MPower mail system. To edit or create a multimedia mail message, the user can access the HP MPower mail editor (composer) by selecting Mail Msg from the elm screen. This will provide the screen shown in Fig. 1b.

To read or view a mail message, the user would select one of the messages in the message list and then press Read Msg menu item from the elm screen shown in Fig. 2. The selected mail message will appear in the HP MPower viewer window for reading.

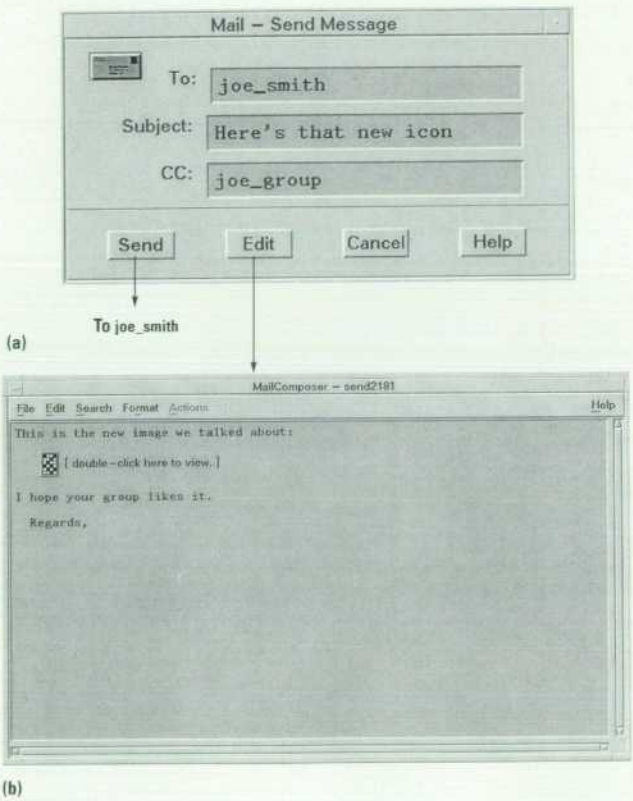


Fig. 1. (a) The dialog box that appears after a mail message is dragged and dropped on the front-panel mail icon. Selecting the Send button will send the message to its destination. (b) Selecting the Edit button produces the mail composer screen.

HP MPower Mail System Components

The main components of HP MPower mail include the HP-UX elm mail user agent, a multimedia editor, several shell scripts, a standard multimedia file format and supporting software, and HP VUE actions and file types. Fig. 3 shows a simplified diagram of some of the main HP MPower mail components responsible for providing the user interface actions described above.

With the exception of vuemime, which among other things handles the encoding and decoding of multimedia data, the

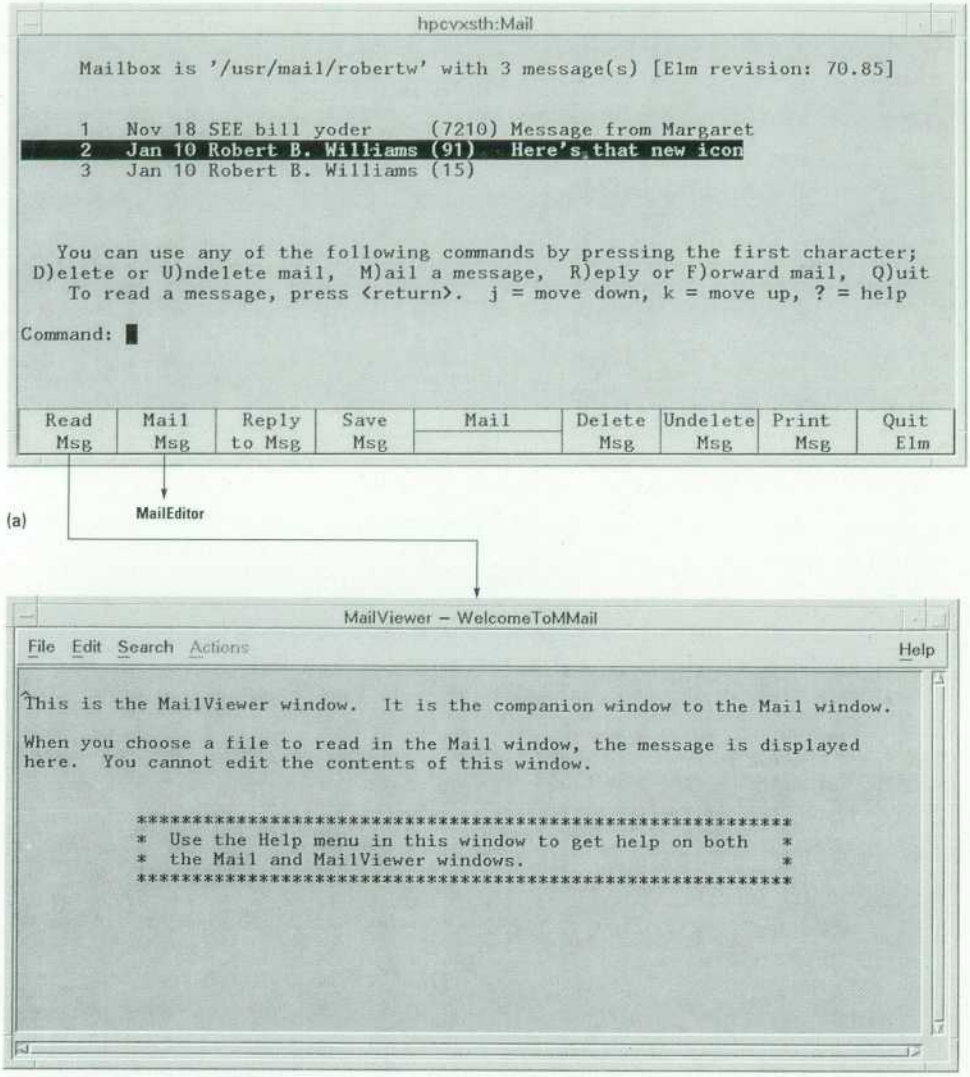


Fig. 2. The windows produced by just clicking the mail icon. The top window contains elm and the lower window contains the HP MPower viewer.

components shown in Fig. 3 that send or receive data require no specific knowledge about dealing with multimedia data. For example, when a media-rich file is sent to sendmail, the file is treated like any other file going onto the network. Another example is when the composer or viewer needs to render a multimedia object (e.g., play an audio file or draw an image), the media icon is mapped to the process (or actions) capable of handling the particular media object by the HP VUE action database.

Vuepad. This is an enhanced version of the HP VUE editor. It provides two modes of operation in the HP MPower mail system: composing (creating a multimedia mail message to send) and viewing (reading a multimedia mail message). Vuepad is described in detail later in this article.

HP VUE Action Database. When a particular file type is selected the HP VUE action database provides the mechanism for invoking the appropriate action associated with that file type. For example, when the user double-clicks on an icon representing an audio file, the HP VUE action database tells vuepad that the audio player should be invoked to play the contents of the file.

MIME, Vuemime, and Metamail. To handle the interchange and storage of different types of data, HP MPower uses an internet standard known as MIME (Multipurpose Internet Mail Extensions). MIME is an extension to the basic internet mail standard RFC 822, which specifies the format for internet text messages. MIME defines the format for multipart, multimedia mail messages. Vuemime and metamail are utilities that provide support to HP MPower for MIME data. Vuemime provides a single interface for HP MPower components using MIME data, and metamail is a public-domain utility that we modified to act as a MIME filter. In this role metamail decodes and flattens MIME messages from non-HP MPower sources and encodes messages going to the mail transport agent sendmail. Vuemime is the only HP MPower component that communicates directly with metamail. MIME, vuemime, and metamail are described in more detail later in this article.

Mail User Agents. HP MPower has two mail user agents, elm and a shell script called mmsend. Elm is the main user agent, providing the usual elm capabilities of viewing, printing, saving, or deleting incoming or stored mail, replying or forwarding mail, and originating mail. Elm is the standard mail

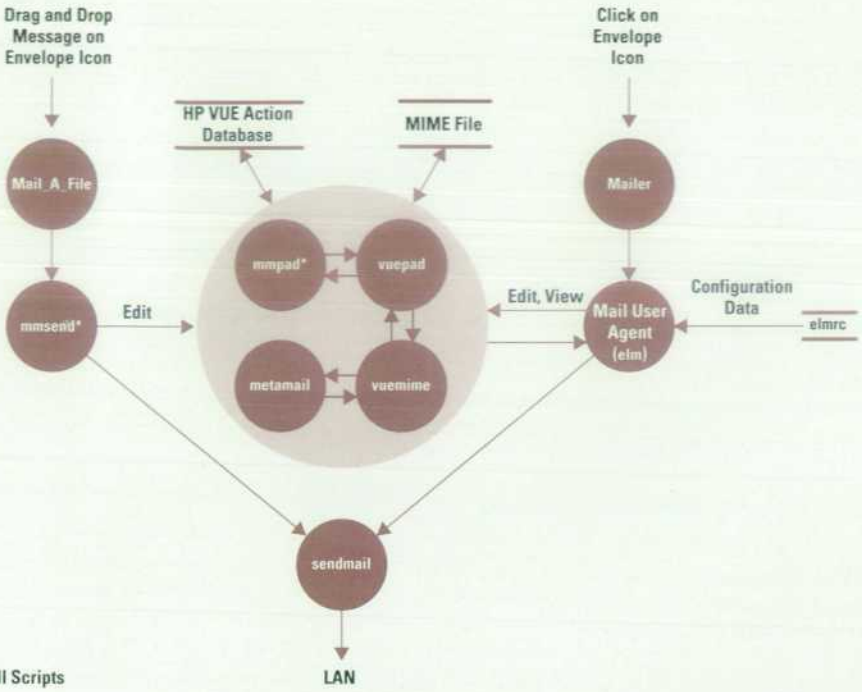


Fig. 3. Some of the main components contained in the HP MPower mail facility.

processor that is shipped with every HP workstation. This greatly reduces the support load and avoids duplication of development effort. Unfortunately it also put us in the position of offering a keyboard-based mail agent amidst a large collection of applications with graphical interfaces. Access to elm is through clicking on the mail icon, producing the screens shown in Fig. 2.

One of the obvious changes to elm's normal interaction model is the tools used to view and edit mail. In the HP MPower environment, since we have to deal with mail that may contain nontextual items, the multimedia composer (editor) and viewer in vuepad are used to edit and view mail. This is done by modifying the user's elmrc (configuration) file so that instead of invoking vi or more for editing or viewing a file, HP MPower's editor or viewer are invoked. Most other user customizations of elm behavior in the elmrc file are preserved.

The second mail user agent, mmsend, provides an interface for drag and drop mailing of multimedia objects. Dropping a file or files on the mail icon causes the software to invoke an HP VUE action that indirectly calls the mmsend shell script. Mmsend drives the processes that are responsible for displaying and interacting with the screens shown in Fig. 1. The mmsend script also invokes the processes that convert multimedia files into MIME format. When the user selects the Send button from the mail dialog box shown in Fig. 1, mmsend invokes the mail transfer agent sendmail to dispatch the message over the network.

Mail Transfer Agent. The primary mail transfer agent in the current version of HP MPower is sendmail, the HP-UX facility for sending mail over the internet. Although there are gateways from sendmail SMTP (Simple Mail Transport Protocol) to X.400 and OpenMail, these gateways are not supported because they have not been enhanced to understand MIME on the SMTP (outbound) side.

Instant ignition systems (see page 17) usually do not have sendmail enabled. To overcome this the HP MPower installation scripts set up a default sendmail.fc and alias database. This is only done if sendmail is not already enabled. It does not address sendmail connectivity issues in complex or restricted environments. Ideally, sendmail would be turned off on miniclient systems because the mail agent runs on the server with the rest of HP VUE.

Printing. Multimedia printing involves decomposing the MIME formatted file into its parts and invoking the HP MPower print action (HP SharedPrint) on each of the parts. HP SharedPrint is described in the article on page 44. Invoking HP SharedPrint for mail is handled by the shell script mmprint.

Several types of multimedia cannot be printed (e.g., audio files). This is handled by a print action for the unprintable file that maps to the special action NONE.

Since printing actions happen on the client (HP SharedPrint forces this), vuemime and its database mimetypes must exist on the client as well as the server. This has implications for installations adding multimedia types.

Multimedia Editor

An important aspect of a multimedia mail facility is the ability to compose and view documents containing both text and nontextual items. In the HP MPower environment this ability is provided by a specially evolved version of the HP VUE text editor, vuepad.† All of the multimedia editing and viewing facilities available in HP MPower are built on this new version of vuepad.

The development of this component was constrained by many different factors. The schedule was set to allow the

† Unless stated otherwise references to vuepad in the rest of this article will be referring to the new version of the program.

introduction of the product to coincide with the release of new workstation computers. This ensured good public exposure for the product, but limited the amount of time and the number of engineers available for product development. The schedule and staffing level provided a strong motivation for using the functionality of existing products rather than wholesale development of new components.

Vuepad Modes. The vuepad program provides two modes. One is the HP MPower mail composer (mentioned previously) for creating and editing multimedia documents. The other mode is the HP MPower mail viewer (also mentioned previously) for viewing or listening to multimedia documents. The viewer is simply a read-only version of the composer. These facilities are accessible from the HP MPower mail icon or the HP VUE panel edit icon. The activation of either of these two modes is determined by the arguments passed to the vuepad program when it is invoked from the mmpad script shown in Fig. 3.

The HP MPower mail composer takes files created by the various media editing tools such as the HP SharedX Whiteboard or the audio editor and allows the user to incorporate the files generated by these tools into a document containing text. The final output of a message created by the composer is in a format compatible with the MIME standard. This approach required no modifications to any of the media editing tools and no changes to the core email application (elm). Use of the MIME message format helps provide some interoperability with other mail systems and gives us access to an already well-documented and carefully defined structure for messages.

Fig. 1 shows that the vuepad editor provides the composer and viewer with a visually appealing "iconic" view of nontextual data. The vuepad editor ensures that a user's normal editing actions function as the user expects in the presence of nontextual items. It also ensures that the operations available for manipulating nontextual items are clearly visible.

Modifications to Vuepad. The required modifications to the old vuepad editor included adding the ability to invoke a filtering program for reading and writing multimedia data. The choice to do this filtering in a separate program was primarily driven by the need to develop the filtering functionality in parallel with the enhanced editor, and a desire to use different development programming languages for the editor and filter.

The filter is the vuemime program mentioned earlier. When vuepad is reading a multimedia file, vuemime strips out the media data, writing each media object to a separate temporary file. The media data is replaced within the original data stream by a special character sequence indicating that a media object existed in that location. This character sequence, known as a tag, contains the name of the temporary file that holds the media data. This file is used by vuepad and the HP VUE file-typing facilities to determine the graphical icon to display in place of the media object. When vuepad writes to a multimedia file, vuemime inspects the file for any tags and replaces the tags with the actual media data and the necessary MIME header information.

It was necessary to augment or override some of the internal functions of the OSF/Motif text widget that vuepad is based on. Before the widget draws a line of text, vuepad checks to see if the text corresponds to part of a media icon. If it does, then vuepad draws the necessary portion of the icon. If the text does not contain part of an icon, then the text widget is allowed to render the whole line of text. Whenever the widget writes or reads data to or from either the OSF/Motif clipboard or the X primary selection window (a clipboard that can hold one item at a time), vuepad runs the data through the vuemime filter program to reinsert or strip any embedded media data as required. When the user selects a portion of text for an editing operation, vuepad tracks the selected region to provide meaningful highlighting of any selected media icons. Vuepad also observes all deletions and additions to the document to accurately track the position of the media objects within the document. Special care is also taken during the spell checking and formatting operations. The spell checking code must exclude the rather cryptic media tag data from the text sent to the spell checking program, and the formatting code has to do its work while preserving the relative locations of the media icons.

Compatibility. The behavior of vuepad is controlled by X Window System resources and command line options that allow vuepad to behave identically to the previous nonmedia-enhanced version of the HP VUE editor. The HP VUE file typing† and action database mechanisms were used to speed development and to provide consistency with HP VUE's file manager appearance. The HP VUE action database provides a simple means for invoking an appropriate action associated with any particular file type. Use of the action database in vuepad ensures consistency with the default action accessed from the file manager for a particular file type.

Composer Features. The HP MPower multimedia composer presents a flat, scrollable view of a document. This allows the user to access any and all document parts rapidly with no enforced sequential ordering. The media objects are embedded in the document, that is, the actual media data is copied into the resulting file rather than having links maintained to the original data. This ensures that the recipient of a message not only has immediate access to the data, but also results in larger messages and freezes the data at the time of composition. The media objects can be embedded at any location within the document, providing significantly more flexibility than the "attachment" model used by many mailers. The attachment model maintains links to other parts of a document.

To incorporate a media object into a document being composed, the user can either use the Include dialog facility of vuepad or drag the object from a file manager view and drop it on the composer's window. If the user wishes to create a new media object while composing a message, the appropriate editing tool (e.g., the audio editor or the image editor) must be used to create the object within that editor. After the media object is created and saved in the file system, vuepad's Include dialog or drag and drop facilities can be used to include the new object in the message being composed.

As shown in Fig. 1, the composer displays icons in place of all nontextual portions of the document. This results in a reasonably attractive appearance and good user recognition because of the similarity with the HP VUE file manager

† The HP VUE file typing mechanism provides the capability to define classes of files. For example, it can define all files ending with .tif to be of class TIFF.

iconic view. The user can either double-click with the mouse, or press the **Return** key to activate the currently selected icon. This provides a simple and quick means of viewing or playing the media item and allows easy access from the keyboard for those users who prefer not to move their hands to the mouse.

The composer also provides an Actions menu item which is sensitive whenever a single media item is selected. This menu contains two items. The first item is Open, which duplicates the functionality of double-clicking on the selected icon. The second item is Save As, which allows the user to save the selected media object to another file separately, with no MIME structure. The Save As action is also the default "open" action of the unknown data type. The unknown data type allows users to pass arbitrary binary data through the mail system undisturbed.

The current implementation of the composer does have some limitations, but a foundation has been built upon which to improve and add features to the current version.

Multimedia Data Extensions

As mentioned previously, HP MPower supports the interchange and storage of several different types of multimedia data contained in a single message or a file via a format conforming to an extension to internet mail known as MIME (Multipurpose Internet Mail Extensions). This multipart, multimedia support is implemented in HP MPower by two utilities: metamail and vuemime. Metamail is a public-domain, sample implementation of a full-featured MIME agent. Vuemime is a MIME filter that was created specifically for HP MPower and HP VUE to simplify generation and manipulation of multipart, multimedia data from the various HP MPower components.

We selected MIME for its strong support within the internet mail community, and we found it to be also useful outside the mail domain. The HP MPower environment recognizes files structured in compliance with the MIME specifications and will deal with them transparently for editing and printing. The composer allows the user to insert nontextual objects into any document being edited, and the resulting file will be saved in MIME format. The MIME message format is also used for cut and paste operations between editing windows. This allows the user to treat editing of mixed-media data in the same manner as plain text. Unfortunately, the mail composer is currently the only application that understands MIME data in cut and paste operations.

This section first provides some background on MIME, describes metamail in the context of HP MPower, and then discusses vuemime in some detail.

MIME Background. Since 1982 the standards that form the basis for internet mail have been defined by RFC 822 and the SMTP (Simple Mail Transport Protocol) defined by RFC 821.† RFC 822 was intended to specify a format for text messages and, as such, did not explicitly allow for inclusion of multimedia messages such as audio or image data. In particular, RFC 822 defines a message as consisting of two parts: a header and a body. The header consists of a series of specific field names and field values followed by a blank line that marks the end of the header and the beginning of the body. The body is restricted to relatively short lines (1000 characters) of seven-bit ASCII characters which cannot exceed a certain length. Users who wish to include nontextual data have to convert the data to seven-bit ASCII before submitting the data to their mail user agent or mail program.

RFC 1049 attempted to rectify some of the deficiencies of RFC 822 by defining a header field and a content type that marks the entire message body as being a certain type of data (e.g., text, audio, video, etc.). In the absence of a content-type field, the body was assumed to be U.S. ASCII text, as before. Although RFC 1049 has been used by several implementations, it is not without problems. The most severe problem is its total lack of support for multipart mail. RFC 1049 allows a message body to be specified as containing something other than text, but only one such thing.

RFC 1341, or MIME, generalizes and extends RFC 1049 in several ways. Most important, it defines a new content type called *multipart*, which can be used to encapsulate several body parts within a single RFC 822 message body. It also goes far beyond RFC 1049 in explicitly describing the set of allowable content types by defining a subtype mechanism for content types that includes provisions for addressing standardized encoding of non-ASCII character sets. It should be noted that RFC 1341 is an extension to, rather than a revision of RFC 822 in that it defines these new features (including text of unlimited line and overall length, characters sets other than ASCII, and multifont messages) within the confines of RFC 822.

The new header fields and content types defined in the MIME extension are described on the next page.

Metamail. Metamail is a public-domain, sample implementation of a MIME agent that was designed to function as a back end for an existing mail user agent. It can be incorporated into virtually any mail reading (or bulletin board) program (e.g., xmail, xmh, elm, etc.), enabling it to become a multimedia reading interface. Metamail knows how to parse a structured MIME message, flatten any hierarchy of nested messages, decode the various parts, and optionally, dispatch the appropriate handlers or viewers for the different parts. The commands used to dispatch the handler or viewer for each content type are specified in one or more mailcap (configuration) files, which allow a great deal of flexibility in adding and configuring handlers. As a viewer, metamail can be thought of as a multimedia counterpart to the ASCII paging tool, more, except that it enlists the aid of additional handlers and viewers when paging through a message in a linear fashion.

In HP MPower we needed more flexibility when composing, viewing, printing, and sending mail so we did not incorporate metamail directly in our mail user agent (elm). Instead, we use the enhanced version of vuepad described above as our mail viewer and composer and we delegate metamail to the role of a MIME filter or preprocessor and postprocessor. In this role, metamail serves primarily to simplify the digestion and generation of MIME compliant messages by other HP MPower components. Specifically, metamail is used to flatten potential message hierarchies and to encode and decode each message part according to its encoding scheme. On the input side (mail receiving), for example, if we receive a MIME message from a non-HP MPower sender, we pass it to

† Each internet standard is defined by one or more standards each known as a Request for Comment, or RFC.

MIME Header Fields

The MIME extension (RF 1341) to the basic internet mail standard RFC 822 created the following new header fields:

- MIME Version. This field is used to specify a version number that declares a message conformant with the MIME standard
- Content Type. This field is used to specify the type and subtype of data in the body of a message and to specify the complete encoding of such data
- Content Transfer Encoding. This field is used to specify auxiliary encoding applied to data to allow it to pass through mail transports having data or character set limitations
- Content Identifier and Content Description. These fields are used to further describe data in the body of the message.

Content Types

MIME enumerates precisely seven valid content types and requires that any additions to this set be specified in a new, similarly formal document. This restriction is a major change from RFC 1049, which allowed for much freer definition of new content types. Instead, the new mechanism for extensions is to define new subtypes of established content types. In general, implementors are required to register new subtypes with the Internet Assigned Numbers Authority (IANA) to avoid name conflicts. (An exception is private subtypes beginning with the letter X, which can be used freely and without registration.)

The seven defined content type values are:

- Text. This is the default content type. The default subtype is plain text. This content type has a charset attribute that has the default value us-ascii.
- Image. This content type is for still images. Subtypes are image format names (e.g., image/gif and image/jpeg).
- Audio. This content type is for audio information. Subtypes are audio format names. For example, audio/basic denotes single channel 8000-Hz μ-law audio data.
- Video. This content type is for video frames. Subtypes correspond to video format names such as video/mpeg.
- Message. This content type is used to encapsulate an entire RFC 822 format message. For example, it can be used in forwarding or rejecting mail. The standard defines two subtypes of message: message/partial, which can be used to break a large message into several pieces for transport so that they can be put back together automatically on the other end, and message/external-body, which can be used to pass a very large message body by reference, rather than including its entire contents.

  It should be noted that a message with a message content type can contain a message that has its own, different content-type field, meaning that the message structure can be recursive.
- Multipart. This content type is used to pack several parts, of possibly differing types and subtypes, into a single RFC 822 message body. The content-type field specifying a multipart type also includes a delimiter, which is used to separate each consecutive body part. Each body part is itself structured more or less as an RFC 822 message in miniature, possibly containing its own content-type field to describe its type. Subtypes of multipart types are specifically required to have the same syntax as the basic multipart type, guaranteeing that all implementations can successfully break a multipart message into its component parts. An expected use of multipart subtypes is to add further structure to the parts and to permit a more integrated structure of multipart messages among cooperating user agents.
- Application. This content type is for most other kinds of data that do not fit into any of the above categories, such as list servers, mail-based information servers, and PostScript.™

A separate part of the content-type header field can be used to convey supplemental information that may be either optional or required, depending on the content type. Such parameters are given in keyword = value format, and are used, for example, to convey information about character sets for text objects. Thus, the default message type for internet mail can be given a MIME content type of:

Content-type: text/plain; charset=us-ascii

Content Transfer Encoding

If internet mail transport (SMTP, as described by RFC 821) is ever upgraded to permit arbitrary binary data of unlimited line length in message bodies, the issue of encoding a message for transport will go away. However, even those who advocate such changes to SMTP generally recognize that they will be slow in coming. In the interim, there is wide perception that a standard mechanism for encoding arbitrary binary data for mail transport is needed.

The content transfer encoding header field can be used to specify the encoding technique used to render binary data in short lines of seven-bit data. After much debate, the working group settled on two encodings, which may be used interchangeably. One of them, the base-64 encoding, encodes each three bytes of binary data as four bytes of 7-bit data, using a base-64 alphabet selected for maximum portability across SMTP implementations, including ASCII to EBCDIC gateways. The other encoding scheme, quoted-printable, is a less efficient representation that preserves nearly all 7-bit ASCII characters as themselves. It is expected that base-64 will be preferred for genuine binary data, while quoted-printable will be preferred for data that is largely U.S. ASCII, but has scattered non-ASCII characters within it. In particular, this may be the preferred encoding for textual email in the national-use variants of ASCII, ISO 8859-X.

If the content transfer encoding field appears in the RFC 822 message header, it refers to the body of the message. If it appears in the header area of one part of a multipart message, it refers to the body area of that part only. The content transfer encoding field is prohibited when the content-type field has a value of multipart or message. This is necessary to prevent nested encodings.

metamail, which filters it by flattening any nested message hierarchies and decoding each message part. The result is a simplified MIME template that can be easily dealt with by other HP MPower components. On the output side (mail sending) metamail is used mainly to encode message parts for handling by the SMTP transport used by our mail transport agent sendmail. While we can handle nested messages on the input side, we only generate flat, single-level messages on the output side.

The official metamail documentation and software, including a draft of RFC 1341, can be found in the pub/nsb directory on thumper.bellcore.com.

Vuemime. In HP MPower, interaction with MIME messages is not the sole domain of the mail user agent (elm) and its viewer and composer (vuepad). Since MIME is used as the storage format for all multipart, multimedia data, printing and sending MIME data as well as composition and viewing have to be performed independently of elm. Printing and sending, for example, can be done by simply dropping a MIME file on the HP VUE front-panel printer or mail icons without invoking elm or vuepad.

Vuemime was created specifically for HP MPower components to provide a single interface to MIME data and, as such, vuemime is the only HP MPower component that directly enlists the services of metamail. Essentially vuemime can be viewed as a higher-level MIME filter which, in addition to providing the type of filtering performed by metamail, provides intermediate formatting that is easily digested by various HP MPower components. What has resulted, after analysis of the needs of various HP MPower components, is essentially five levels of formatted data which can be viewed as stages in a file's morphosis from raw data (level 0) into a fully formatted

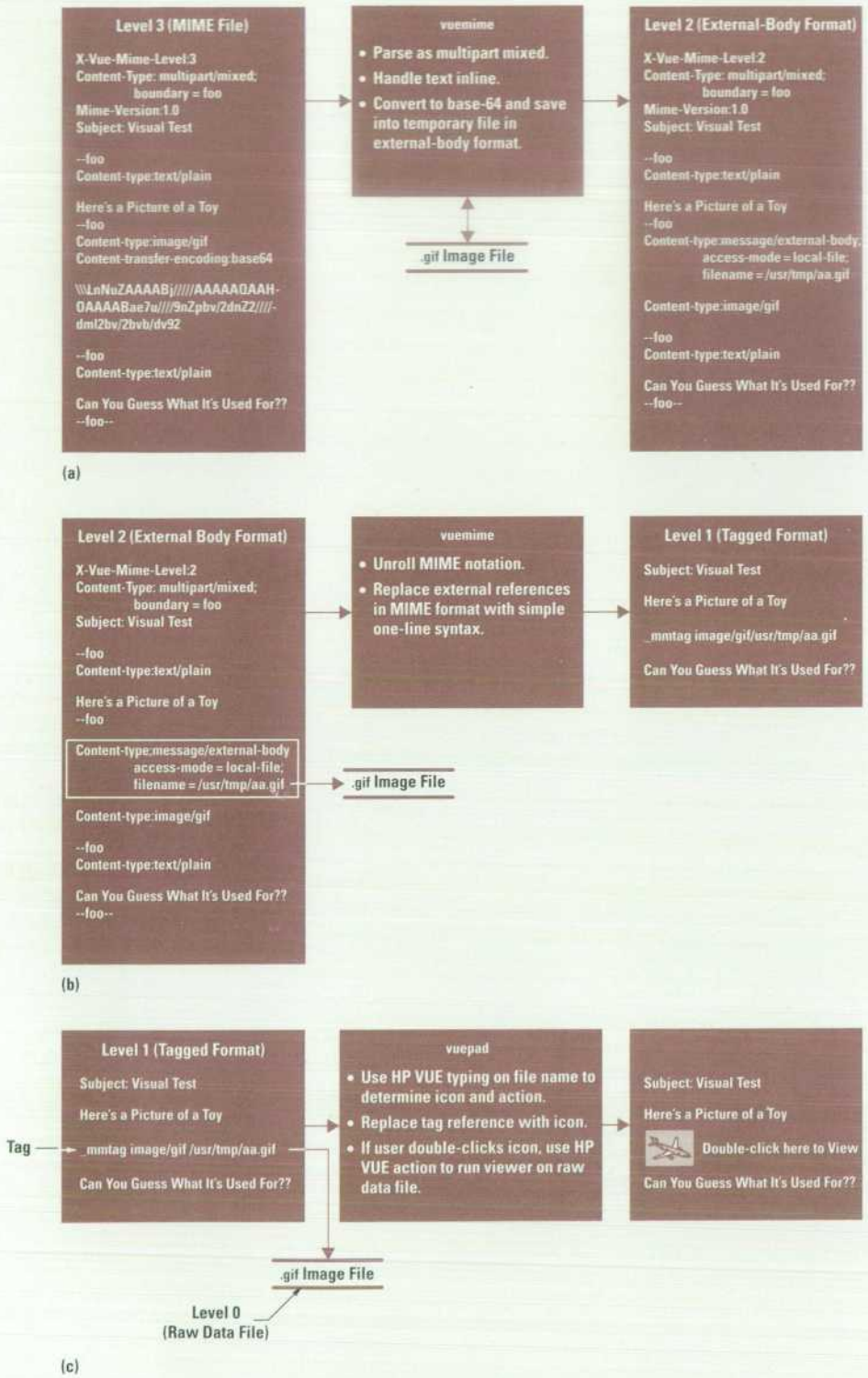


Fig. 4. Processing an incoming message. (a) Level-3 file format and the actions of vuemime to move from a MIME format to a level-2 file format. (b) Level-2 file format and the actions performed by vuemime to transform the data to a level-1 representation. (c) Level-1 file format representation and the actions taken by vuepad to display textual and nontextual parts of the message.

MIME message ready to be handed off to the SMTP mail transport.

Fig. 4 shows the steps involved in transforming an incoming message in MIME format (level 3) through the various levels of formatted data to a tagged (level 1) format. For an outgoing message, vuemime will take a tagged message and generate a self-contained MIME message that can be passed to the mail transfer agent.

The five levels of formatted data shown in Fig. 4 are defined as follows:

- Level 0. This is raw data consisting of ASCII text and nontextual information such as image, audio, and video data.
- Level 1. Data at this level, which is used by vuepad and the mmprint action, is in a special tagged format that contains only ASCII text, summarized mail heading information, and special tag lines representing multimedia objects. The tag lines indicate the multimedia type and a path to a file containing the raw data. See the level-1 file representation in Fig. 4c.
- Level 2. This is an external-body file format. The format at this level is very similar to level 1 except that MIME header and control lines have been added to make it fully MIME

compliant. All multimedia objects are still external (and raw) but are now identified by Content-type: <message>/<external-body> and Content-type: <type>/<subtype> lines. See the level-2 file representation in Fig. 4b.

HP MPower does not currently send messages in level-2 file format, but this level could be useful in at least two situations. First, level-2 format could greatly reduce the size of a message by not sending the actual contents of something like a video clip. In this case, the video information is not sent until and unless the receiver wishes to view the video. The other important use of level-2 file format is to provide a "hot link" to the current version of some data. For instance, a message might contain an external-body reference to some data that is updated hourly. The receiver of the message would see valid data at the time the message is read, instead of the data as of the time the message was sent.

- Level 3. This is a MIME file. All references to external multimedia data files are replaced by encoded data. See the level-3 file representation in Fig. 4a.
- Level 4. This is a MIME file with an outgoing mail address and header information (from the mail user agent) prepended to it. At this level the data file is ready to be transported to the network services.

Vuemime accepts command line options to transform data in either direction and between any levels from 0 to 3. Level 4 is generated solely by the mail user agent (mmsend) for outgoing mail. The HP MPower component (vuepad) simply indicates to vuemime on the command line the level of the file being transformed and the level to which it is to be formatted.

Vuemime generates only base-64 encoded data when going from a level-2 (or lower-level) file to a level-3 (MIME) file even though it can accept other types of encoded data when going from a level-3 file to a lower level. Also, vuemime only generates flat multipart and mixed level-2 and level-3 messages even though it can accept and digest (via metamail) nested level-3 messages generated by a non-HP MPower MIME system.

Mimetypes. Transformation to and from raw data by vuemime is controlled by a file called mimetypes. The mimetypes file is a cross between metamail's mailcap files and HP VUE's file-type definition files. Table I shows the contents of the mimetypes file supplied with HP MPower 1.0.

Vuemime uses the data in the first column to map level-0 (raw data) files to the content type for files that are being transformed to a higher level. The third column is the content-type value inserted in files of level 1 and above. For level-1 files and above that are being transformed to level-0 files, the column 2 entry associated with a particular content type is used to determine the file name extension to be applied to the new raw data file.

Conclusion

Our multimedia email project is an example of a highly leveraged effort that combined existing blocks of functionality, some modifications, and new "glue" to meet a market need quickly.

Table I
Contents of the HP MPower Mimetypes File

Search Pattern for Raw Data Files	File Extension for New Raw Data Files	Content-Type Value
.*\.xwd	%s.xwd	image/x-xwd
.*\.xd	%s.xd	image/x-xwd
.*\.tif	%s.tif	image/x-tiff
.*\.tiff	%s.tiff	image/x-tiff
.*\.eps	%s.eps	image/x-eps
.*\.pcl	%s.pcl	image/x-pcl
.*\.gif	%s.gif	image/gif
.*\.jpg	%s.jpg	image/jpeg
.*\.jpeg	%s.jpeg	image/jpeg
.*\.pm	%s.pm	image/x-xpm
.*\.xpm	%s.xpm	image/x-xpm
.*\.bm	%s.bm	image/x-xbitmap
.*\.xbm	%s.xbm	image/x-xbitmap
.*\.bmf	%s.bmf	image/x-bmf
.*\.ps	%s.ps	application/PostScript
.*\.txt	%s.txt	text/plain
.*\.au	%s.au	audio/basic
.*\.l16	%s.l16	audio/x-Linear16
.*\.l8	%s.l8	audio/x-Linear8
.*\.lo8	%s.lo8	audio/x-Linear-8Offset
.*\.wav	%s.wav	audio/x-microsoft-RIFF
.*\.snd	%s.snd	audio/x-NeXT
.*\.u	%s.u	audio/x-MuLaw
.*\.al	%s.al	audio/x-ALaw
.*\.Z	%s.Z	application/x-compress
.*\.tar	%s.tar	application/x-tar
.*\.unk	%s.unk	application/octet-stream

Acknowledgments

The multimedia mail team offer a special thanks to Gabe Beged-Dov, a former HP engineer, for discovering the leverage possibilities in the MIME internet standardization efforts.

A Fast and Intuitive Online Help System

The HP Help System provides application developers with the tools to create and integrate rich online help information into their OSF/Motif-based applications.

by Michael R. Wilson, Lori A. Cook, and Steven P. Hiebert

With the growing complexity of today's UNIX*-operating-system applications, and the desire to improve usability, media-rich information is becoming more pervasive in today's computing environments. Users expect some base level of online help to be provided from within the applications they are using. They expect online information to be intuitive and graphical, with growing expectations of direct audio and video support and interactive capabilities.

The HP Help System is a good start towards providing multimedia online information that is both fast and intuitive. It has become the standard online help system within HP and is used extensively by the HP VUE and HP MPower products and many other OSF/Motif-based products.

Background

The current HP Help Developer's Kit is the second attempt by the HP VUE program to deliver an application help system for everyone. The first version, HP Vuehelp 2.0, while satisfying some of the HP VUE requirements as an application help system, failed to meet application developers' requirements for features, performance, and ease of integration.

One of the design goals for the HP Help System was to deliver a complete solution to developers for creating, integrating, and shipping rich online information with their OSF/Motif-based application, while keeping its presence (system resource use) to a minimum. There is a very fine line between providing the rich set of features that our customers require and maintaining our performance objectives. In 95% of the cases in which the issue of features versus performance came up, performance won.

Based on the knowledge and insight our team gained in doing the first version of the HP Help System, and a willingness to make radical changes in our second release, we basically started from scratch. We knew that our next-generation help system needed to provide a competitive set of base features and functionality that developers require, while producing little or no end-user-visible performance degradation to the hosting application.

The Help Developer's Kit

The HP Help Developer's Kit is a complete system for developing online help for any OSF/Motif-based application. It allows authors to write online help that includes graphics and text formatting, hyperlinks, and communication with the application. It provides a programmer's toolkit allowing developers to integrate this rich online help information into their client application. The help dialog widget (Fig. 1) serves as the main display window for the HP Help System. A second, lighter-weight help widget (quick help dialog) is also available in the toolkit.

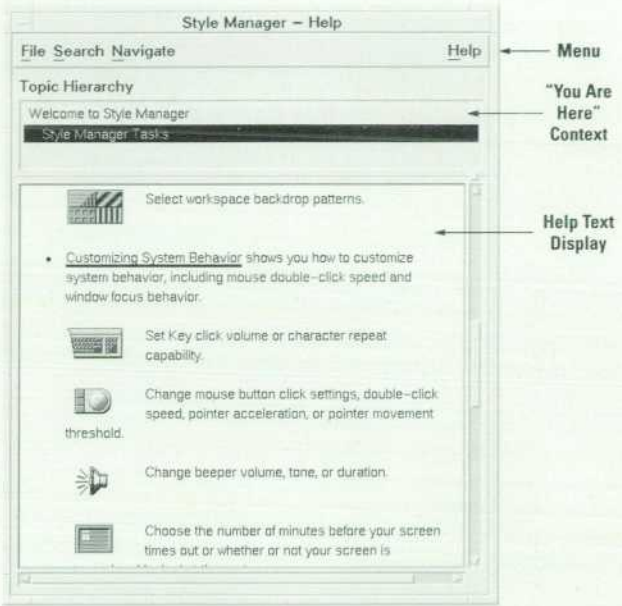


Fig. 1. Help dialog widget.

Following is a list of the components supported in the developer's kit:

- For Authors
  - The HP HelpTag markup language. This is a set of tags used in text files to mark organization and content of online help information. HP HelpTag is based on SGML (Standard Generalized Markup Language).
  - The HP HelpTag software. This is a set of software tools for converting the authored HP HelpTag files into run-time help files that contain the text for the help messages displayed in the help widgets.
  - The HelpView application. This is a program that allows an author to test a newly developed online help facility.
- For Programmers
  - The Xvh programming library. This library provides an application programming interface for integrating help windows into an application.

- A demonstration program. This is a simple example that shows how to integrate the HP Help System into an OSF/Motif application.

General Packaging Architecture

An online help system needs to feel like it is part of the host application to the end user, not an appendage hanging off to the side. For developers to leverage a third-party help system, it must be delivered in such a way as to provide easy and seamless integration into their application. Furthermore, the effort and overhead of integrating and redistributing this help system along with their application must be minimal, while at the same time meeting the application's and end-user's requirements for help. Users should feel as if they have never left the application while getting help.

During our initial prototyping of the current HP Help System, we kept stumbling on the same two key issues: performance (how to make the system light and fast) and packaging (how to make it easy to integrate into any OSF/Motif-based application and redistribute with that application). Our initial help system suffered greatly in both of these areas. HP Vuehelp 2.0 was server-based, large, slow, and dependent on the HP VUE desktop. Any application using our help services had to run within the HP VUE desktop environment.

We addressed these two issues with our current release and ended up with a very different package. To fix the performance problems of slow startup times and heavy server-based implementation, we started copying functionality from the help server into the client via a linked-in help library. While prototyping this architecture, we quickly realized that we were duplicating services. However, we were also getting much better performance from the new client code. At that point we realized that if we moved everything to a help library we could fix our two biggest problems: performance and packaging.

By moving to a help library and removing our hard-wired dependencies on the HP VUE desktop and by bundling our product with HP's User Environment Developer's Kit, we cleaned up our previous dependency problems with HP VUE. Now developers can embed help directly into their application and ship a single executable that includes both. Developers only have to link their OSF/Motif application with the HP Help System library and they are ready to go. All dependencies on external system and desktop services have been removed. Fig. 2 shows an overview of our old and new help system architectures.

Following are some of the advantages we gained by moving from our initial HP Vuehelp 2.0 server-based implementation to an embedded client-side architecture.

- Integration Advantages:
  - OSF/Motif-based application program interface (simple to use for developers familiar with OSF/Motif)
  - Complete application control over the help system dialog management including creation, destruction, caching, and reuse
  - Smooth transition into the help system via consistent resource settings between the application and the help system (e.g., same fonts and color scheme and quick response times)

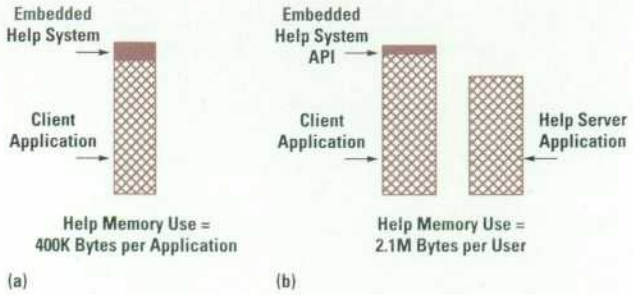


Fig. 2. Two different approaches to integrating online help into an application. (a) Client side embedded implementation using HP VUE 3.0 help. (b) Help server implementation using HP VUE 2.0 help. Our initial prototypes revealed that the client side embedded solution is better in terms of performance (rendering time) and memory use.

  - Immediate support for a tightly coupled application-to-help system environment (e.g., application-defined and processed help controls such as hypertext links)
  - Application-specific customization of the help system.
- Packaging Advantages:
  - Eliminates installation and version problems (Since applications are not sharing the help services, the contention between older or newer software versions does not exist.)
  - Eliminates distribution problems. (The help system is linked directly into the host application, tested and shipped as one executable.)
- Performance Advantages:
  - Requires less overall system resources (RAM and disk) for a single application using the help services
  - Provides much faster initial response time when displaying help requests.

Integration Concepts, Practices, and Mechanisms

As mentioned in the previous section, the run-time help facility is made up of a collection of help dialog widgets and files containing online help information. The help widgets are linked directly into the client application via the help library libXvh.a* and instantiated by the client to display help information. While the help dialogs serve only as vehicles for displaying online help information, standard OSF/Motif, Xlib, and X toolkit (Xt) services provide the glue to integrate the dialogs into the application.

The online help files in the HP Help System are called help volumes. These volumes contain the text for the help topics that are displayed in the dialog widgets. The following files, or volumes are used by the HP Help System:

- volume.hv. This is the master help volume file accessed by the HP Help System when a user makes a request for help information. Information stored in this file is used to access the actual help topics stored in the help topic (.ht) files.
- volume.hvk. This is a keyword index file for the master help volume.
- volumeNN.ht. These are the help topic files, where NN represents file numbers (00, 01, 02 ...). If there are no chapter elements in a help volume, only a single topic file (e.g., volume00.ht) will exist. These are the files containing the help text.

The relationship between these files is shown in Fig. 3.

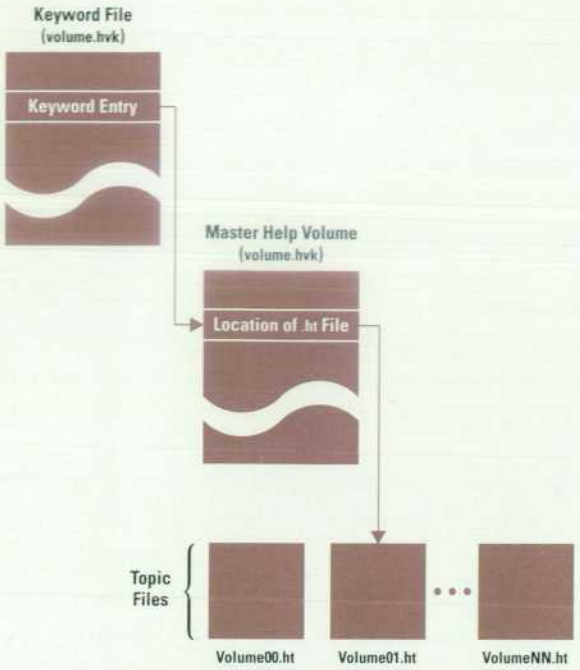


Fig. 3. The HP help file system.

There are two levels of integration with respect to the HP Help System: integrating help into an application and integrating a help-smart application into an HP VUE or HP MPower desktop environment.

Integrating Help into an OSF/Motif Application

Developers have many degrees of freedom with respect to how much or how little help capability they include in an application. If an application and its help information have very loose ties, there may be only a handful of topics that the application is able to display directly. In contrast, the application could provide specific help for nearly every object and task in the application. This requires more work, but it provides potentially greater benefits to the end user.

Help Dialogs. Two help widgets are supported in the help library: quick help dialog and general help dialog. They both support the same text, hypertext, and graphics display capabilities, but differ with respect to the remainder of the user interface. The quick help dialog (Fig. 4) is a very simple help dialog intended for displaying small blocks of text to the user. Quick help dialog is best used for handling things like error messages, version information, and object and item definitions.

The general help dialog, which is the most commonly used help dialog, has a few more user interface features than the quick help dialog widget. Most notably, the Topic Hierarchy list (see Fig. 1), which appears just above the help text display area, indicates the location of the current topic in the help topic hierarchy. The general help dialog also provides various navigational aids to assist the user in moving about the online help information space.

Standard Xt Paradigm. The programmer interacts with the help dialogs in the same manner as any other OSF/Motif widgets used by the application. The two types of help dialogs are defined by the following two widget classes:

- XvhQuickHelpDialogWidgetClass (for quick-help dialog)
- XvhHelpDialogWidgetClass (for general-help dialog).

Nearly every attribute of the help windows including the volume name and topic identifier are manipulated as widget resources. For instance, to display a new topic, an XtSetValues() call is made to set the volume, location identifier, and help type resources.

Creating Help Entry Points

Each help topic that can be displayed directly as the result of a help request is called an entry point. That is, if there is at least one way to get directly from the application to a help topic, then that help topic is an entry point into help. Help menus and buttons in the application are the basic help entry points for an application. The types of help requests that provide entry points into the HP Help System are contextual help and item help.

Contextual Help. Contextual help provides help information about the item on which the selection cursor is positioned. Contextual help information provides users with information about a specific item as it is currently used. The information provided is specific to the meaning of the item in its current state. For example, suppose a user is running an application that uses an options widget that has four options. If the user requests information on the widget while it has option 1 selected, the user will get help information on the option widget in the context of its option 1 setting.

A selection cursor, which is a visual cue, enables users to indicate with the keyboard the choice with which they want to interact. It is typically represented by highlighting the choice with an outline box.

The OSF/Motif user interface toolkit, through its help callback mechanism, provides direct support for contextual help entry points. When a valid help callback is added to a widget, and the user presses the help key (**F1**) while that widget has the current keyboard focus (e.g., selection cursor), the widget's help callback is automatically executed.

From within the help callback, the application has the opportunity to display some help topic based on the selected

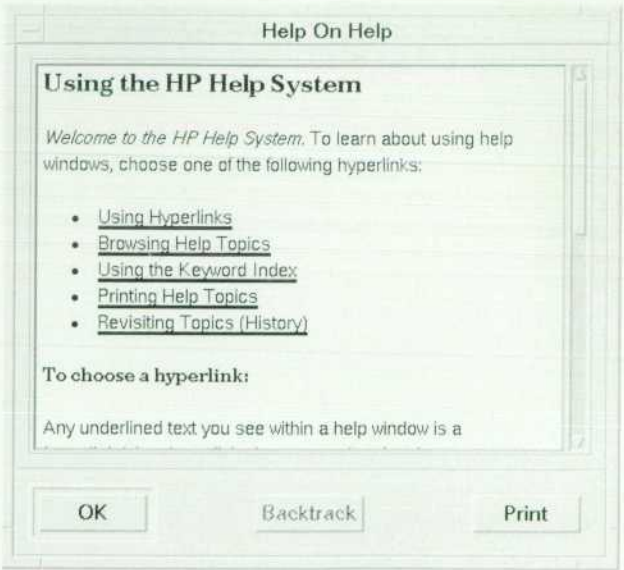


Fig. 4. Quick help dialog box.

widget, or the application could dynamically construct some help information based on the current context of the selected item.[1,2]

Any level of granularity can be applied when adding help callbacks to an application's user interface components. They can be added to all the widgets and gadgets within the application dialogs, the top-level windows for each of the dialogs, or any combination in between.

If the user selects **F1** help with the selection cursor over a widget or gadget that has no help callback attached to it, the OSF/Motif help callback mechanism has a fallback mechanism for providing more general help. The help callback mechanism will jump to the nearest ancestor that has a help callback assigned and invoke that callback. The theory is that if there is no specific help on that widget or gadget, then it is better to provide more general help than none at all. Application developers are responsible for adding their own help callbacks to the user interface components in their application. OSF/Motif sets these callbacks to NULL by default.

Item Help. Item help allows users to get help on a particular control (e.g., button, menu, or window) by selecting the item with the pointer. Item help information should describe the purpose of the item for which help is requested and should tell users how to interact with that item. An item help request does not provide context sensitive information like the current state of the selected item.

Item help is usually accessed via an application's Help menu under the On Item menu selection. Once selected, the cursor is replaced with a question mark cursor. The user can then select the item of interest.

The HP Help System API utility function, XvhReturnSelectedWidgetId() assists developers in providing item help within their application. This function provides an interface for selection of a component within an application. XvhReturnSelectedWidgetId() returns the widget identifier for any widget in the user interface that the user has selected via the pointer.

At any point while the question mark cursor is displayed the user can select the escape key (**ESC**) to abort the function call. If the user selects any item outside the current application windows the proper error value will be returned.

Once XvhReturnSelectedWidgetId() returns the selected widget identifier, the application can invoke the help callback on the identified widget or gadget to process the selected item. From the help callback the application can display some help topic based on the selected widget or dynamically construct some help information based on the current selected item.

Integrating a Help-Smart Application

There are no restrictions regarding where run-time help files are installed. However, a suggested practice is to package a help volume in a separately installable file set so that the system administrator can place them on a system file server. This will save local resources and make the help information available to a larger number of users. The default configuration is for the run-time files to be installed with the rest of the application's files.

Another important step in installing help files is registration (or symbolic links to help volumes). The registration process

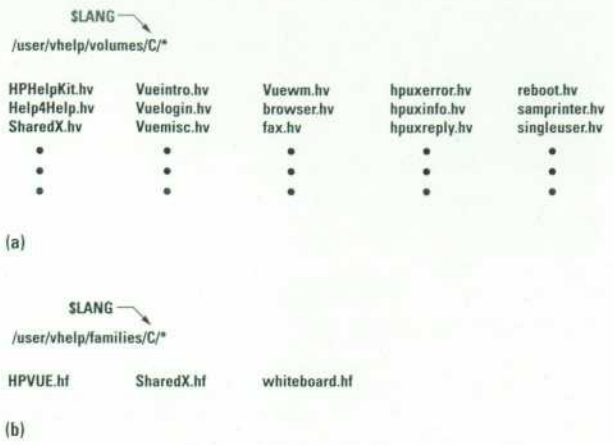


Fig. 5. Help directory contents. (a) A portion of a directory containing help volumes. (b) A directory containing product families.

enables two important features of the HP Help System: cross-volume hyperlinks and product family browsing.

Registering a Help Volume. After the run-time files have been installed, a help volume is registered by creating an HP-UX* symbolic link to the help volume's volume.hv file. The link is created in one of the directories that the help system searches for help volumes (Fig. 5a). For most help volumes, the appropriate place for the link is in the /usr/vhelp/volumes/$LANG/ directory, where $LANG represents the language of the help volume being registered. The default language for help files is usually English.

Registering a Product Family. A product family is a group of help volumes belonging to a particular product. To register a product family, a help family file (product.hf) must be created with the rest of the product's help files. The family file is registered by creating a symbolic link to the product-name.hf file. For most products, the appropriate place for the link is in the /usr/vhelp/families/$LANG/ directory (see Fig. 5b).

Family files can be read with the helpgen program (part of HP VUE), which creates a special help volume that lists the families and the volumes within each family installed on the system.

Access to Help Volumes

The HP Help System has a simple, yet extensible mechanism for transparent access to help volumes installed on the desktop. It supports both local and remote access to help volumes and works with any number of workstations. The only dependencies required are the proper help registration as discussed above, NFS services (i.e., remote systems mounted to the local server), and proper configuration of the help environment variables discussed below.

When an application creates an instance of a help widget the XmNhelpVolume resource can be set using either a complete path to the volume.hv file, or if the volume is registered, using the base name (i.e., the file name without the .hv attached) of the volume. When using the base name, the help system searches several directories for the volume. The search ends when the first matching volume.hv file is found. The value of the user's $LANG environment variable is also used to locate help in the proper language (if it's available).

The environment variable XVHHELPUSERSEARCHPATH specifies the user search path for locating help volumes. Whenever this resource is set to a relative path, the user's default home directory will be prepended to the value contained in the XmNhelpVolume resource. The default value used when the environment variable is not set is:

```
vhelp/%T/%L/%H.hv:\
vhelp/%T/%H.hv:\
vhelp/%T/%L/%H:\
vhelp/%H:\
vhelp/%T/C/%H.hv:\
vhelp/%T/C/%H:
```

where:

%L is the value of the $LANG environment variable (C is the default)
%T is the type of file (volume or family) being searched for
%H is the help volume specified.

These path names are illustrated in Fig. 5.

The environment variable XVHHELPSYSTEMSEARCHPATH specifies the system search path for locating help volumes. The default value used when this environment variable is not set prepends the string /usr to the strings given above. For example, vhelp/%T/%L/%H.hv:\ becomes /usr/vhelp/%T/%L/%H.hv:\. Note that the user search path defined above takes precedence over the system search path.

Run-Time Help Volumes

The flexibility and power of this help system is largely placed in the author's hands. With the HP HelpTag markup language and a creative author, very different and interesting approaches can be taken with respect to presenting information to the end user. Documents can be organized in either a hierarchy with hyperlinks referencing the children at any given level, or in the form of a network or web, with a linear collection of topics connected with hyperlinks to related topics (see "Information Models for Online Help" on page 92 for more about these document organizations). It's up to the author to explore the many capabilities with respect to producing effective online help information for a particular system.

Help Volume Structure. A help volume is a collection of related topics that form an online book. Normally, the topics within a volume are arranged in a hierarchy. Developers usually create one help volume per application. However, for complex applications or a collection of related applications several help volumes might be developed. Topics within a help volume can be referenced by unique location identifiers that are assigned by the author. It is through these location identifiers that help information is referenced in the run-time environment.

Help Volume Authoring. Online help is written in ordinary text files. Special codes, or tags, are used to mark up elements within the information. The tags form a markup language called HP HelpTag. The HP HelpTag markup language defines a hierarchy of elements that define high-level constructs such as chapters, sections, and subsections, and low-level elements such as paragraphs, lists, and emphasized words (Fig. 6). The text files created by authors and any associated graphics files are compiled using the HP HelpTag software

```
<!entity versionGraphic FILE "bike.bm">

<metainfo>

 <title>Helpdemo (Sample Application)
 <copyright>

    <graphic entity=versionGraphic><term nogloss!Helpdemo!, Version 1.0

    <image>
    &copy; Copyright Hewlett-Packard Company 1992
    All Rights Reserved.
    </image>

    !!This program is for demonstration purposes only! !!

 <abstract>This online help volume is used with the 'helpdemo' program
       that demonstrates how to use the HP Help System in an OSF/Motif
       application.

</metainfo>

<hometopic>Helpdemo Help Information
       <idx!introduction!

This is the home topic. This is the top level of your helpdemo
help information.

Choose one of the following links to find out more about the helpdemo
program.

<list bullet>
 * <xref chap1ID>
 * <xref chap2ID>
</list>

<chapter id=chap1ID>An Application Help System

<list bullet>
 * <xref onApplicationMenu>
 * <xref sampleBtnOne>
 * <xref sampleBtnTwo>
</list>


<s1 id=sampleBtnOne>Button One Help

Here's the help text for our sample button one.

<s1 id=sampleBtnTwo>Button Two Help

Here's the help text for our sample button two.

<s1 id=onApplicationMenu>Introducing Helpdemo

This is the topic displayed by choosing On Application from the Help menu.

<chapter id=chap2ID> Controlling The Application

The following links demonstrate how the HP Help System can control the
hosting application.

<list bullet>
 * <link hyperlink="100" type=AppDefined> Move the window UP </link>
 * <link hyperlink="101" type=AppDefined> Move the window DOWN </link>
 * <link hyperlink="102" type=AppDefined> Move the window LEFT </link>
 * <link hyperlink="103" type=AppDefined> Move the window RIGHT </link>
</list>

Note: The text contained in the brackets "<>" is the tags that form the
      HP HelpTag markup language.
```

Fig. 6. A sample HP HelpTag volume that is distributed as part of the HP Help Developer's Kit.

to create run-time help files (Fig. 7). These run-time files (or volumes) are installed with the application and accessed when the user requests help.

Help Volume Compilation. The HP HelpTag compilation process performs the following tasks in generating a compiled run-time help volume:

- Syntax validation
- Conversion from the authored format to run-time format
- Location identifier map generation
- Topic compression
- Topic hierarchy map generation.

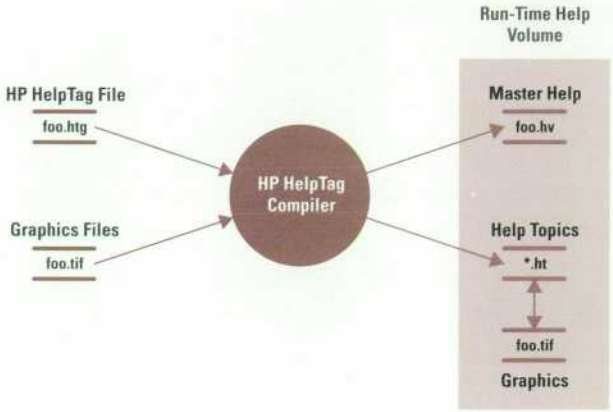


Fig. 7. The HP HelpTag compilation process.

While creating the help volume compilation process we developed techniques for improving the performance (speed and size) of our system. Our objectives were to create a run-time file format that supported very quick access (three seconds from request to display) for any requested help topic contained within a help volume, while at the same time keeping the overall size of the volume as small as possible. Topics with graphics can sometimes be slower than three seconds.

To solve the quick access problem, a scheme was devised that takes the physical hierarchy of authored topics within the volume and flattens the whole volume. During this process a jump table is generated that lists each topic name, the file that topic resides in, and the offset into the file that represents the beginning of the topic. With this flattened format and the jump table, topics can be quickly retrieved for display. The jump table is stored in the volume.hv file in X resource format, and parsed using standard xrm function calls in Xlib.

Help Volume Compression and Decompression. The size of compiled run-time help files was a real problem in the first prototypes. The solution we developed to solve this problem involves compressing the topics in the help topic (*.ht) files. This task was added as the last step in the compilation phase of an authored help volume.

The help data files are compressed on a per-topic basis. That is, each help file is read to determine the start and end points of each topic in the data file. This information is used to step through the data file, extracting each topic in turn, writing it out to a temporary file, and using the HP-UX utility compress(1) to compress the file. Unless a special option (–f) is given to compress, the utility will not compress a file if the file does not actually shrink. (The HP Help System is able to read uncompressed files.) After executing the compress utility, the resulting file is checked for a .Z extension to determine if compression has actually taken place.

If the topic is compressed, a null byte followed by three bytes making up a 24-bit number holding the size of the topic in bytes is written to a new version of the help data file followed by the compressed topic. If the topic is not compressed, it is simply copied to the new help data file. In either case, a new version of the index file (volume.hv file) is updated with the new starting offset of the topic based upon the size of the previous topics.

Since no uncompressed topic will ever start with a null byte, the leading null byte in compressed topics serves as a "magic number" to indicate to the run-time help viewer that the topic is compressed. The 24-bit size value written to the file in a fixed byte order allows help files to work on machines with varying byte orders in machine words and is used by the decompression algorithm in the run-time help viewer to determine when to stop expanding a topic.

After the entire help data file has been compressed, the new versions of the help data file and index file are moved to replace the old versions.

When the compression algorithm was first prototyped, we were pleasantly surprised by the results. The new scheme saved over 40% in disk use per volume while creating no end-user-visible performance degradation in rendering time. We use the standard HP-UX compress(1) command, which is called from the HP HelpTag program to handle the compression, and an embedded library version of uncompress for decompression at display time. We determined that the lack of performance degradation in using compressed topics is partly because most of the help topics are very small and the embedded decompression function enables fast retrieval.

Graphics Compression and Decompression. Support is also included for compressed graphics including X pixmaps, X bitmaps, and X windows, as well as JPEG compressed TIFF images, which are supported by the HP Image Library. See the article on page 37 for more about the HP Image Library. In most cases the best results, both for performance and disk use, are gained by using JPEG compressed TIFF images.

Help Dialogs

From the developer's perspective the help dialog is seen as a single widget. Widgets are created, managed, and destroyed as one single object. In reality our help widgets are not one monolithic help entity, but two separate very distinct components: the text display engine component and the widget component. The text display engine component as seen from the developer's perspective is the region in the help widget that renders the text and graphics, provides hyperlink access, and performs dynamic formatting of the text. The widget component consists of the OSF/Motif code that makes it a widget and the remainder of the user interface including topic hierarchy, menu bar, and supporting dialogs (print, keyword search, and history).

The Help Widget. By building the help dialogs as true OSF/Motif widgets (customized or part of a toolkit) and exposing this as our help API, we immediately gained an industry-standard format for our help functions. The syntax and use model for programmers is the same for every OSF/Motif widget created. This makes it very easy for developers familiar with OSF/Motif programming to understand and use the help widgets.

The OSF/Motif-based API supports the various controls we need to manage an instance of our help dialog. Through standard OSF/Motif and X toolkit and Xlib functions, programmers have direct access to the help dialog resources and can manipulate these values as they see fit. Developers can add callbacks via XtAddCallback(), set and modify resources via XtSetValues(), manage and unmanage the dialogs

via XtManageChild and XtUnmanageChild, and free the resources when done via XtDestroyWidget().

The Display Area. Text formatting on the display allows different types of text. The author can specify dynamic text that will format or reformat text according to the window size. The author can also specify static text that will not be reformatted to adjust to different window sizes. For dynamic text a sequence of two or more spaces will be compressed into a single space and internal newlines (**Return**s or **Enter**s) will be changed into a space. For static text all spaces and internal newlines will be honored.

Help system users demanded the ability to resize help windows. While vertical scrolling is accepted as necessary when help topics are longer than the available screen space, horizontal scrolling is not. Therefore, dynamic reformatting is a must.

Foreign Language Format. Text formatting for foreign languages placed some special demands on our display area text formatting widget.

- European (8-bit) rules. For most European languages (including English), breaking a line of text at spaces is sufficient. The only other line-breaking rule applied is with hyphens. If a word begins or ends with a hyphen, the hyphen is considered a part of the word. If the hyphen has a nonspace before and after it, it is considered a line breakable character and anything after it is considered safe to place on the next line.
- Asian (16-bit) rules. For Asian language support, breaking a line of text on a space is unacceptable since some 16-bit languages do not break their words with spaces (i.e., Japanese and Chinese, but not Korean). With the Japanese language, the characters are placed one after another without any word breaking character because each character is considered a word. There is also the concept that certain characters cannot be the first character on a line or the last character on a line. English (8-bit) characters can be mixed with 16-bit characters.

Given these considerations, the following line-breaking rules for 16-bit languages are used:

1. Break on an 8-bit space.

2. Break on a hyphen if the character before and after the hyphen is not a space.

3. Break on an 8-bit character if it is followed by a 16-bit character.

4. Break on a 16-bit character if it is followed by an 8-bit character.

5. Break between two 16-bit characters if the first character can be the last character on a line and the other character can be the first character on a line.

Rather than hard code the values of those Japanese characters that can't start or end a line into our help system, a message catalog system is used. This provides a general mechanism for any 16-bit language. All language localization support people are required to determine which characters in their language cannot start or end a line and localize the appropriate native language support (NLS) file. The file /usr/lib/nls/%T/%L/fmt_tbl.cat contains the values of those characters that are used for rule 5 of the line-breaking rules. If this file does not exist for a given language, rule 5 is ignored and line breaking will occur between any two 16-bit characters.

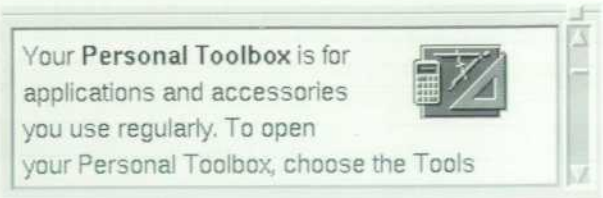


Fig. 8. Text flowing around a graphic.

Even using these line-breaking rules, sometimes the lines are still too long to fit in the available space. For this case or when static text pushes the boundary of the display area, the help system gives up and displays a scroll bar so that the user can see the information without having to resize the window.

The help system uses the same routines for processing English or multibyte documents. These routines determine what line-breaking rules to use based on the $LANG environment variable and the character set used in the document.

If a document specifies an ISO-LATIN1 character set, the display engine does not bother looking at rules 3 to 5 or using multibyte system routines. Only when the document and the $LANG environment variable specify a 16-bit character set are these rules and routines used. This minimally impacts the access and rendering time but allows the same binary to be used in the United States, Europe, and Asia.

Flowing Text. The help system display area also provides the ability to flow text around a graphic. This is seen as a space saving measure and a highly desired feature. The graphic can be placed on the left side or the right side of the display area and the text occupies the space to the side of the graphic. If the text is too long and does not fit completely in the space beside the graphic, the text wraps below the graphic. Fig. 8 shows an example of this graphics formatting technique.

Graphic Manipulation. Complaints about the text-only nature of HP Vuehelp 2.0 strongly demonstrate the truth of the adage "one picture is worth a thousand words." Therefore, the help system tries to allow as many standard graphic formats as possible. During the design of the help system the question was, which ones to allow? Eventually, it was determined to allow standard X graphics (plus the new X pixmap) and TIFF image formats. The reason for selecting the X graphic formats is that they are supported by standard Xlib routines for accessing graphics files, and the reason for choosing TIFF format is that the HP Image Library supports TIFF format. The formats supported by the help system include:

- X bitmaps
- X window files
- X pixmap files
- TIFF 5.0.

Graphic Compression. While JPEG compression schemes are common for use with TIFF files, no compression was being used with the X graphical formats. After numerous complaints from authors about how much space Xwd files

(continued on page 88)

WYSIWYG Printing in an X Application

The HP Help System provides several features that allow authors to specify different fonts in various sizes and combine them with graphics bitmap illustrations. These capabilities and others created special challenges when it came to WYSIWYG (what you see is what you get) printing. The HP Help System developed some sophisticated techniques for handling WYSIWYG printing.

What Is WYSIWYG?

Our market research uncovered a strong demand for WYSIWYG printing without a clear definition of what WYSIWYG really meant. One obvious interpretation is the screen dump that accurately represents on paper what is on the screen. This is appropriate for some applications, but prototypes of this approach showed jagged characters crammed into a small 4-by-6-inch window on a large 8½-by-11-inch piece of paper. This satisfies no one. Another approach is to take a printed image and display it on the screen matching line breaks as they appear on the printer. This approach compromises readability on the screen, which is not acceptable for an application providing online information. Fig. 1 shows one interpretation of WYSIWYG printing.

The interpretation of WYSIWYG we chose for our help system is based on the notion that "WYSIWYG is a state of mind." True WYSIWYG is undesirable, but an appropriate illusion of WYSIWYG is the right answer. For the HP Help System, we give presentation on the screen and presentation on the printer separate consideration. The printed page uses the same fonts as the screen (sans serif for the body of the text and serif fonts for titles), but printer fonts are laid out and rendered at 300 dots per inch. The help topics are laid out to fit on the size of the printed page, and each page has page numbers. Thus, line breaks and page breaks on paper do not match line breaks and screenfuls on the display, but this is just what our customers are looking for. Fig. 2 shows a comparison of screen and printer output.

Font Availability

We took care to allow the HP Help System to work with any X server. This was important to us, since we knew our work was likely to be ported to other platforms besides the HP-UX operating system. What is more, the distributed nature of X means that even though the application may run on an HP-UX computer, it can be displayed on a totally different device, such as an HP Vectra PC running an X server under Microsoft® Windows. It is difficult to predict what fonts might be available to the server.

To enable the help system to use fonts available on any display server as well as those built into HP LaserJet printers, we developed a table that maps generic help fonts to specific fonts available on the target server or printer. After having studied a variety of HP documentation, we identified a need for three font families for help messages: a serif proportionally spaced family (like Times), a sans serif proportionally spaced family (like Helvetica), and a serif monospace family (like Courier). Each family contains four treatments: normal, bold, italic, and bold italic. Table I shows these fonts. For special characters, a symbol font is also included. Our discovery of this was no surprise. It corresponds very closely to the 13 typefaces typically found in PostScript™ printers and in the HP LaserJet III printer.

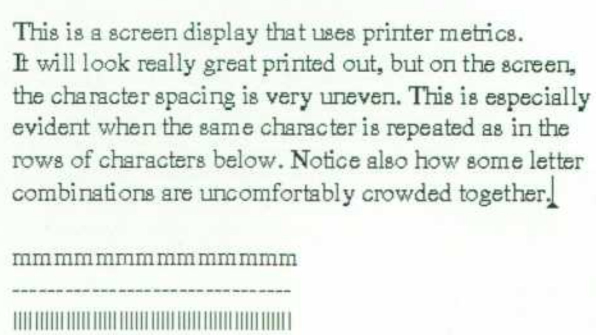


Fig. 1. One interpretation of WYSIWYG printing.

Table I
HP Help System Generic Fonts

Typeface Category	Treatments			
	Normal	**Bold**	*Italic*	***Bold Italic***
Serif	A	**A**	*A*	***A***
Sans Serif	A	A	*A*	*A*
Monospace	A	**A**	*A*	***A***

For screen use, we mapped our generic fonts to fonts included in the standard X distribution, which is available on most if not all X servers. The distribution uses New Century Schoolbook, Helvetica, and Courier fonts. Should these fonts not be available, an administrator can change the mapping table. The table is in an X resource file. For situations in which all the fonts are not available, several generic fonts can be mapped to the same physical font. We did this with Asian languages in which all four typeface treatments are not available. For Japanese, the serif family is Mincho and the sans serif family is Gothic, but all treatments are mapped to the one treatment available.

For HP LaserJet III printers, we used the same scheme to ensure we used the built-in fonts CG Times, Univers, and CG Courier. Thus, the fonts on the printer are not identical to those on the screen, but they are close enough to create a WYSIWYG state of mind to our users. Table II shows various font family mappings.

As an example of how this mapping works consider an application that requests a sans serif bold 12-point font. This request will be honored with Helvetica on a European X server, Univers on an HP LaserJet III printer, and Gothic on a Japanese X server.

Table II
Font Family Mappings to Printer or Screen

Typeface Category	Fonts Available: X Server (European)	HP LaserJet III	X Server (Japanese)
Serif	New Century Schoolbook	CG Times	Mincho
Sans Serif	Helvetica	Univers	Gothic
Monospace	Courier	Courier	Mincho

Use of generic fonts worked so well for us that we are now investigating ways to build this capability into X font servers so that applications can be assured of a reasonably rich set of fonts no matter what the platform or the current locale.

Xlib for Printing

The traditional approach to the implementation of printing in an application is to write the rendering code at a reasonably high level and then write specific drivers that convert from the higher-level code to the page-description language of the printer. With this approach an application might end up with a large number of drivers, one for each specific type of printer.

We addressed printing rather late in the development cycle. By the time we began developing printing support, the rendering code had already been written directly to Xlib, making it difficult to develop a higher-level tool kit as described above. We turned this vice into a virtue by experimenting with a new concept using Xlib functions.

Using some X toolkit and X motif routines we developed a clone of Xlib called **Xvplib**, which has the same functions as Xlib, but generates PCL instead of X protocol. Printing is done through a separate program called **helpprint**, which is called by the help widget. The **helpprint** program uses the same help library (**Xvh**)

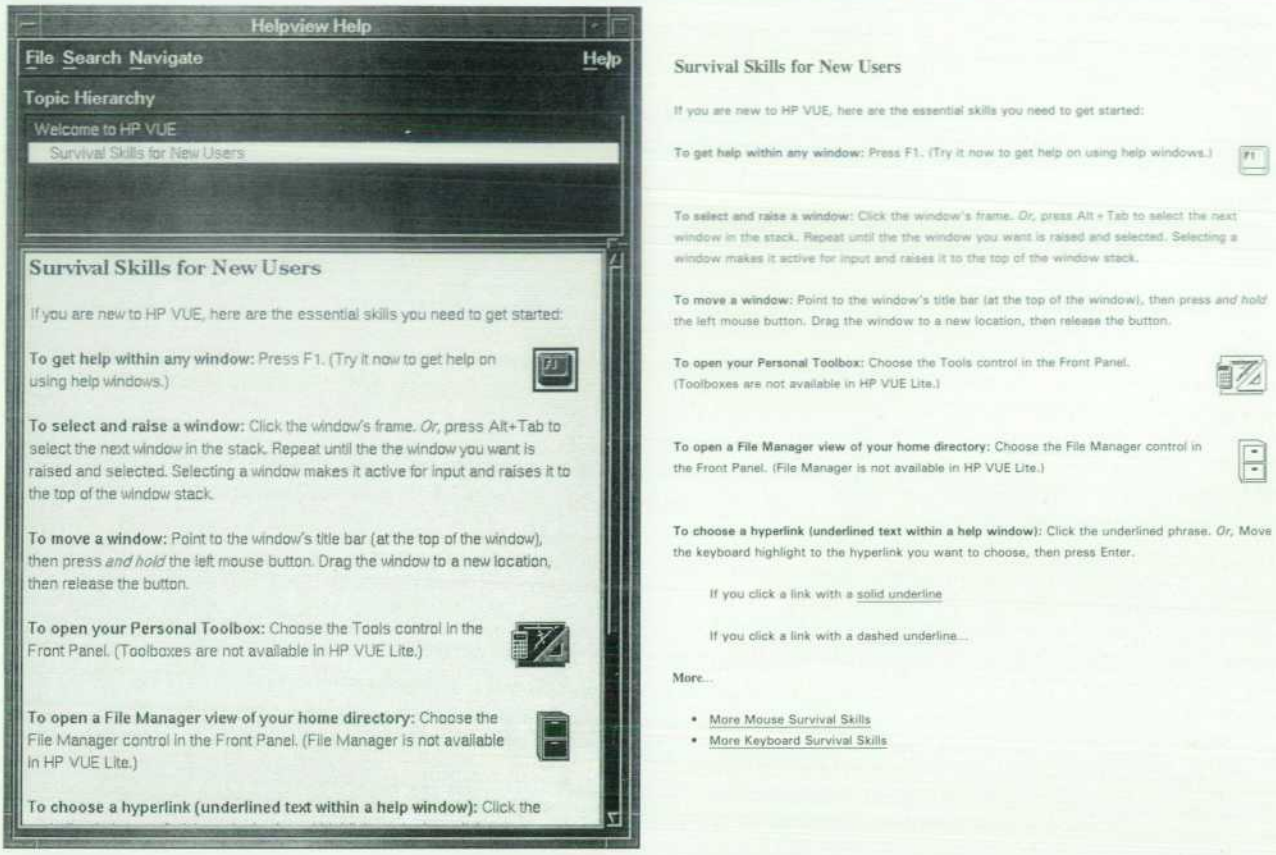


Fig. 2. A comparison between display (left) and printer (right) output from the help system.

as the display program, but links with XvpLib instead of Xlib for the printer driver code (see Fig. 3).

The helpprint application uses the Xlib call XOpenDisplay() to open a printer and then uses the XCreateSimpleWindow() routine to create a window on the print display. This window reflects the margins on the page. The help-rendering library (Xvh) then renders the current help topic into the window. The rendering process performed by Xvh involves retrieving the window size using XGetWindowAttributes(), loading fonts using XLoadQueryFont, and using XDrawImageString() to render text until done. The XvpLib library provides the Xlib functions and generates the PCL code required. Many pixel arguments such as width, height, x, and y are much larger to printer displays than to screen displays because of the 300-dpi resolution of the printer. However, calls such as XTextWidth(), which returns the space required to draw a string, still work fine. The XvpLib library supports both the HP LaserJet II (PCL 4) and the HP LaserJet III (PCL 5) printers. The primary difference between them is the selection of built-in fonts. The match from X to PCL was surprisingly close, but some X calls (such as XORing bitmaps) had to be interpreted liberally. XvpLib supports all 36 X functions used by the help topic library Xvh.

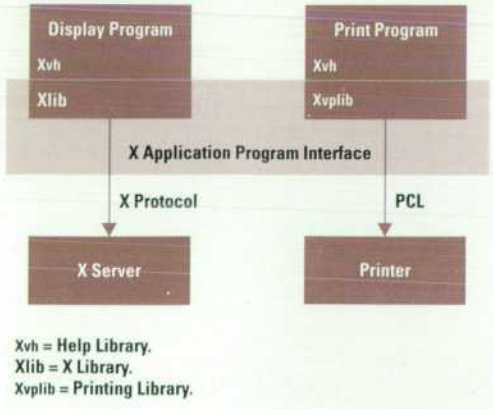


Fig. 3. The architecture for HP help printing drivers for the display and for the printer.

Using X as our print interface worked very well. Only about three lines of code had to be added to the help-rendering library (Xvh) to support printing. Common rendering code for the screen and printer made it easy for us to keep the layout of help topics consistent between screen and printer.

Printing for Special Printers

The printing approach described above works very well with PCL printers. However, HP customers in Japan presented us with special challenges. First, the Japanese language requires support for 16-bit text, and these customers typically have printers that support Asian page description languages such as Canon's LIPS and Ricoh's RPL.

To support such requirements, we took a different approach. Instead of using helpprint to render to the printer, we developed another rendering program called helpprintrst. This program creates an offscreen pixmap the size of a sheet of paper on the X server and and then calls Xvh to render into that pixmap. The print program retrieves the completed page using XGetImage() and sends it to the printing library (XvpLib) to turn the image into a PCL bitmap. Special filters in the print spooler convert the PCL bitmap into a bitmap suitable for the target printer.

Axel Deininger
Engineer/Scientist
Workstation Technology Division

require, the help system was modified to find and access compressed files. The author uses the same HP-UX compression utility mentioned earlier to compress the graphic files. The same internal decompression routine used to decompress topic files is used to decompress the graphic files into a temporary file. After decompression, the help system reads the graphic file as usual.

Using compression on X graphic files can impact access time. For very large graphic images or for a topic that uses many graphics, this impact can be noticeable. The trade-off between speed and disk space is one that the author and application engineer must address.

Color Degradation. Another obstacle encountered with handling graphics was color graphics on grayscale or black and white displays. Having to provide three different images for the three types of displays is not feasible because of disk use. Therefore, the help system has to degrade a color image for grayscale or black and white displays.

For X bitmaps, this is no problem since bitmaps are specified as using the foreground and background colors only. For TIFF images, the HP Image Library forces the image to the proper color set depending on the display type. For X pixmaps, the Xlib routines take the same approach depending on the display. This leaves the X window files to manipulate.

The task is to reduce an Xwd file from a full color image to a grayscale image containing no more than the maximum number of gray colors we have available. This number is kept in a variable called MAX_GRAY_COLORS. The HP Help System uses eight gray colors. The first step in this process is to map each of the X window color pixels to a grayscale luminosity value (① in Fig. 9). This is done using the NTSC (National Television Standards Committee) formula for converting RGB values into a corresponding grayscale value:

$$\text{luminosity} = 0.299 \times \text{red} + 0.587 \times \text{green} + 0.114 \times \text{blue}$$

where red, green, blue are the X window color values in the range of 0 to 255. This creates a grayscale luminosity value in the range from 0 to 255.

The next step is to determine the number of distinct luminosity values used in the image. This involves counting the total number of luminosity values computed above (② in Fig. 9). The idea is to group the grayscale luminosity values across the available gray colors so that the image can be evenly and reasonably rendered.

After determining the actual number of distinct luminosity values used in the image, a further calculation is done to determine which gray color to use for a given group of luminosity values (③ in Fig. 9). If the number of distinct luminosity values is greater than or equal to MAX_GRAY_COLORS, then each gray color will be used at least once. However, if the number of distinct luminosity values is less than MAX_GRAY_COLORS a calculation is performed to spread the color load among the gray colors instead of bunching them together. This provides the contrast in an image.

Finally, when the image is rendered to the display the shade of gray associated with a particular group of luminosity values is mapped to the appropriate display pixel (④ in Fig. 9).

If the system has to degrade the image to black and white, it first calculates the grayscale colors using the luminosity calculation described above. Then it dithers the image using a Floyd-Steinberg error-diffusion algorithm, which incorporates a Stucki error filter.[3]

The end user can force the whole environment including the help system to use grayscale or black and white by setting it via the HP VUE style manager's Color Use dialog. Also, if an image uses more colors than are available in the color map, the help system will automatically degrade the image until it is renderable.

Hard-Copy Support

A critical and difficult requirement of this help system was to provide WYSIWYG (what you see is what you get) printing support to match our text display capabilities. To ensure good performance and not tie up an application just to print out its online help, we implemented the printing mechanism as a separate application. When an end user requests to print one or more topics in the current help volume, the widget code packages the request and performs a vfork(2) to launch the print application.

The help system supports two different print applications: helpprint which is for HP LaserJet printers (i.e., PCL support) and helpprintrst for printing help volumes that contain multibyte characters such as those used in some Asian languages. The helpprintrst command operates just like the helpprint command except that its output does not depend on printer fonts. Instead, helpprintrst creates a page-size graphic image of each help topic.

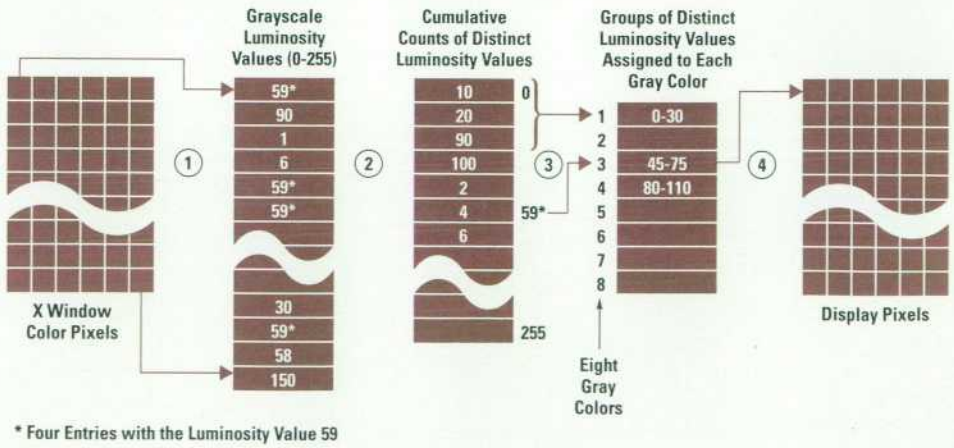


Fig. 9. The process of reducing a full color image to a grayscale image.

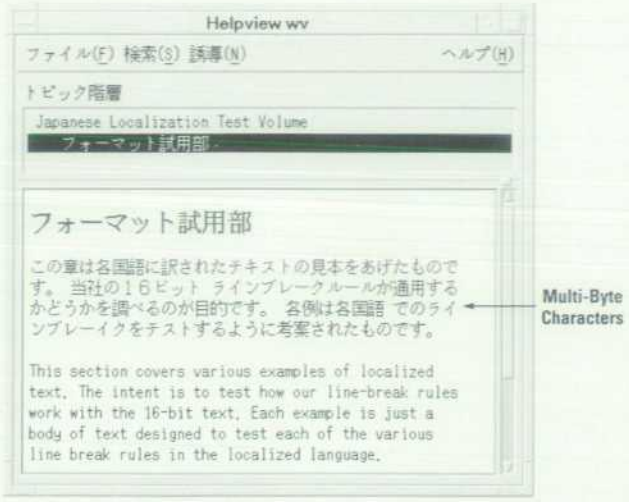


Fig. 10. A localized help window.

The goal in implementing a WYSIWYG printing solution for the help widgets was centered on creating a solution that could solve the printing problem for many applications, not just the HP Help System. Refer to "WYSIWYG Printing in an X Application," on page 86 for a description of the printing solution used by the HP Help System.

Localizability

The HP Help System supports the authoring and displaying of online help in virtually any language. The online help information can be authored and translated in either single-byte or multibyte character sets. All the components within the developer's kit are multibyte-smart and can parse and display the localized information.

The help widgets use the $LANG environment variable to determine which language directory to search to retrieve the requested help volume. For example, if $LANG = Japanese when the request to display help occurs, the widget code will attempt to open the Japanese localized version of that help volume. If one does not exist, then the default version, which is English, will be used.

When an authored help volume is compiled via HP HelpTag, the author sets the proper character set option. The character set information is used at run time to determine the proper fonts to use for displaying the localized help text. The default character set for the HP HelpTag compiler is the one defined for 8-bit character sets by ISO 8859-1. Currently only one character set per volume is supported.

Fig. 10 shows a sample of a localized help window.

Parsing Multibyte Characters. To make a single tool work for single-byte and multibyte character sets without constantly checking for the length of the current character, all characters are converted to wide characters on input. This input conversion is driven by either command line option, on an HP HelpTag entity file, or by the current setting of the locale. All internal processing of characters is done on wide characters with those wide characters being converted back to multibyte characters on output. Single-byte character sets are treated just like multibyte character sets in that the same input and output conversions always take place.

This scheme of doing all internal processing on wide characters has proven to be a very effective means for making one tool work for all languages. The scheme did require implementation of wide character versions of most of the string functions (e.g., strcpy, strlen) but those functions were all quite straightforward to create.

Localizing User Interface Components. The menus, buttons, labels, and error messages that appear in help dialogs also support full localization to native languages. The help dialogs read the user interface component strings from a message catalog named Xvh.cat. Different localized versions are supported by default and included with the developer's kit. For languages not supplied with the help system, the developer must translate the message catalog /usr/vhelp/nls/C/Xvh.msg and then use the gencat command to create the needed run-time message catalog file.

Acknowledgments

The authors wish to acknowledge all those who contributed to the design and development of the HP Help Developer's Kit, including the R&D, learning products, and marketing groups at the Open Systems Software Division in Corvallis, as well as the many other contributors throughout HP. Appreciation to Brian Cripe for his willingness to beat the issues into resolutions, to Axel Deininger for performing coding miracles with respect to WYSIWYG printing, and to the many managers that supported this project through to release: Bob Miller, Rick McKay, and Ken Bronstein. Special thanks goes to Dex Smith for helping our team understand the value of SGML and pushing us in that direction with respect to our product offering.

References

1. *OSF/Motif Style Guide*, Revision 1.2, Prentice-Hall, 1993.
2. *OSF/Motif Programmers Guide*, Revision 1.1 Prentice-Hall, 1991.
3. R. Ulichney, *Digital Halftoning*, MIT Press, 1988, pp. 239-244.

Developing Online Application Help

The primary goal for an application help system is to provide the capability for the end user to get useful help information and get back on task as quickly and successfully as possible.

by Dex Smith

Nearly 2000 online help topics are shipped with the HP MPower product. Throughout the HP VUE 3.0 and HP MPower projects, we've learned a lot about online help and its role with application software.

This paper describes online help and covers many of the issues encountered by application developers and authors. First, we outline the purpose of online help and suggest a use model. Second, two common information models are described, including how each relates to the domain of online help. Next, we discuss what is perhaps the most challenging and controversial topic in online information—navigation. Although this paper won't go into the details of these debates, it does reference some of the concerns and issues. Finally, we examine the roles of developers in a typical project, and we look at how these roles and online help systems may change in the future.

Although much of this material was developed during the design of the HP Help System and includes examples from that product, most concepts and ideas can be applied to any help system. The details of the implementation of the HP Help System are described in the article on page 79.

The Role of Application Help

To understand the purpose of online help, we begin with a prioritized list of goals, called a goal hierarchy.[1] Our goal hierarchy for online application help consists of the following:

1. The user leaves the help system.
2. The user is satisfied with the help information.
3. The application's response to the user's request for help is appropriate.
4. The help information is appropriate, even if the system response is not.
5. The user can pursue additional information.

From the user's perspective, the first two goals are clearly most important. The user wants to get out of the help system and back to work. Obviously, we want the user to return to work with useful information.

To application developers, goal 3 is an important reminder that help is part of the application. A well-designed application will use everything at its disposal (current modes, recent user actions and errors, and other contexts) to determine the best possible response to a user's request for help.

Goal 3 is also important to designers of a help system. It emphasizes that the design must provide the flexibility and power for application developers to integrate online help that is capable of responding as the user expects.

The fourth goal represents the help author's obligation to design and deliver the most appropriate information. It further acknowledges that there may be limitations in the help system, or in the application's use of system features, that the author may need to work around in designing the online information.

Finally, with the fifth goal, if the user's need is not immediately met, the system must allow the user to seek additional information.

All of these goals can be summarized in a single prime directive for an application help system—get the user back on task as quickly and successfully as possible.

The Application Help Use Model

Since the overriding principle for providing online help is to get the user back on task, online help must be easily accessible within whatever applications the user has chosen to accomplish the task. To that end, online help is cast in a supporting role. That is, the help system's job is to function as part of the application with the purpose of making the user successful with the rest of the application.

Keeping the help system tightly coupled with the application improves the user's *perceived proximity* (i.e., keeping the user feeling close to the application). If deviations to get help don't take users far from their current context within the application, they are more likely to stay on task and be productive. Avoiding unnecessary context switches is a fundamental human factors guideline.

The software architecture used to implement the help system doesn't necessarily have to be part of the application. The HP Help System's toolkit allows software developers to embed the help system within the application or to create a standalone help server.

If the help server model is used, special attention must be given to window manipulation and user interface design, so that the user still perceives the help as part of the application.

Handling Help Requests

In its simplest form, the interaction of a help system is a request/response transaction model. A request is made that

represents a need for information, and the system responds by providing information that meets the need.

For HP Help, we have categorized help information retrievals using the terms automatic, semiautomatic, and manual as follows:

- Automatic help. The application determines what help to present and when to present it. For example, an error message is automatic help because the application automatically provides information to the user. (This is sometimes called system-initiated help.)
- Semiautomatic help. The application determines what help to present based on the current context in the application. The user determines when information is presented by making a request for help. The de facto standard gesture for this kind of request is pressing the **F1** key.
- Manual help. The user requests a particular type of help. In this case, the user determines what kind of help and when to deliver it. Most manual help requests are made by choosing a help command from the application's Help menu.

Entry Points into Help

When a help request is made (either by the user or automatically by the application), the system responds by displaying a help topic. Each topic that can be displayed directly as the result of a help request is called an *entry point*. That is, if there is at least one way to get directly from the application to a help topic, then that help topic is an entry point into help.

Any single topic may be an entry point from more than one context within the application. The redundancy of overloading entry points in this way can often be advantageous to the user.[2] For example, a topic on entering numeric data may be appropriate help for many places within a database application.

Styles of Entry Points. Our goal hierarchy and use model has led us to classify two styles of entry points for help topics:

- Quick help. This style presents a focused topic for the current context, intended to meet the user's need and then to be immediately dismissed.
- General help. This style presents a help topic for the given request with the understanding that the user is likely to want additional information.

Comparing the two styles is like contrasting an information booth with a consulting office. At an information booth, contexts are well-defined, questions are generally short, and answers come quickly enough that there's no need to sit down. In seeking the services of a consultant, however, the expectations are different. You want to make good use of your time to learn, see examples, and explore related subjects so you don't have to come back.

An entry point into help is the first thing a user sees when a help request is made. The presentation, user interface, and information content set the user's expectations about how specific the information is to the current context and how quickly it can be dismissed.

For most applications of even moderate complexity, a combination of both styles of entry points is appropriate. The next two sections describe each style in more detail.

Topics that are not entry points can be reached only by navigating from other topics or by other means within the help system. Therefore, the organization and relationships between help topics are important to designing appropriate entry points. The impact of the organization of help topics on navigation is discussed later.

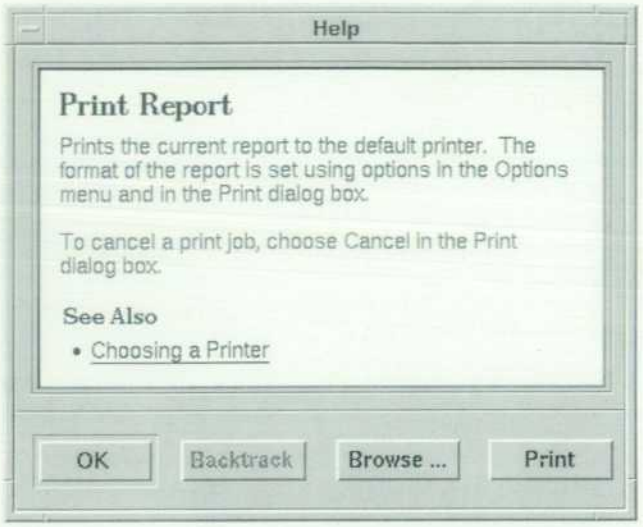


Fig. 1. A sample quick help dialog from the HP Help System.

Quick Help. Quick help entry points are optimized by writing style, presentation, and user interface to get the user back on task with minimal interruption. Quick help should have a very simple user interface and present easily consumable portions of information.

The quick help concept addresses the first goal in our goal hierarchy: User leaves the system. The second goal—User is satisfied with the help information—is up to the designers. That is, the correct entry point must be displayed and it must contain correct and useful information.

Quick help topics typically have a high level of context sensitivity and contain very specific information. By design, the content and presentation of quick help should not encourage the user to explore. Hypertext links (hyperlinks) should be used sparingly, and restricted to closely related topics.

Notice the Browse... button in Fig. 1. This is an example of how goal 5 in our goal hierarchy can be addressed for quick help. That is, the Browse... button provides a path for the user to seek additional information.

Quick help entry points are recommended for automatic and semiautomatic help requests. Some manual help requests may also use quick help, depending on what type of information is being requested. For instance, users expect minimal interruption when requesting "help on version" to get copyright and version information.

Typical uses of quick help include the following:

- Error messages, progress information, and feedback on user actions
- Context-sensitive help (invoked with the **F1** key or Help button in a dialog)
- Spot help (help on a particular item or area on the display)
- Application copyright and version
- Simple help on help.

General Help. General help typically covers the remaining entry points into online help. General help may have a more feature-rich interface for searching and browsing. Typical uses of general help include:

- Application overview
- A follow-up to quick help when quick help isn't enough (This use may follow one or more steps in a progressive disclosure sequence (described below).)
- Tutorials (Some tutorials may want to be more restrictive on navigation and prefer to use the quick help model.)
- Online documentation. (For browsing online information written and organized like a traditional manual.)

A sample general help dialog is shown in Fig. 2.

Progressive Disclosure

Even the best online help implementations are unlikely to provide entry points into help topics that meet every user's need for each possible situation.

So, what if a quick help entry point doesn't fill the user's need? Progressive disclosure sequences are designed to maintain a sense of proximity, while exploring a constrained path of topics that incrementally provide more information for the given context. (This concept is sometimes called progressive revelation or progressively more help.)

Often, a progressive disclosure sequence leads the user from initial "how to" context sensitive help through topics that are more motivational ("why") and eventually to detailed reference information. Generally, these sequences use a simple quick help interface for the first two or three progressions. At the point in the progression when the subject matter will likely lead the user to want to browse, the interface should provide additional browsing capability.

Fig. 3 shows how a quick-help dialog can provide a More button for walking through a progressive disclosure sequence. In each step in the sequence the dialog is reused, until the last step when the More button is renamed Browse and leads to a general help dialog.

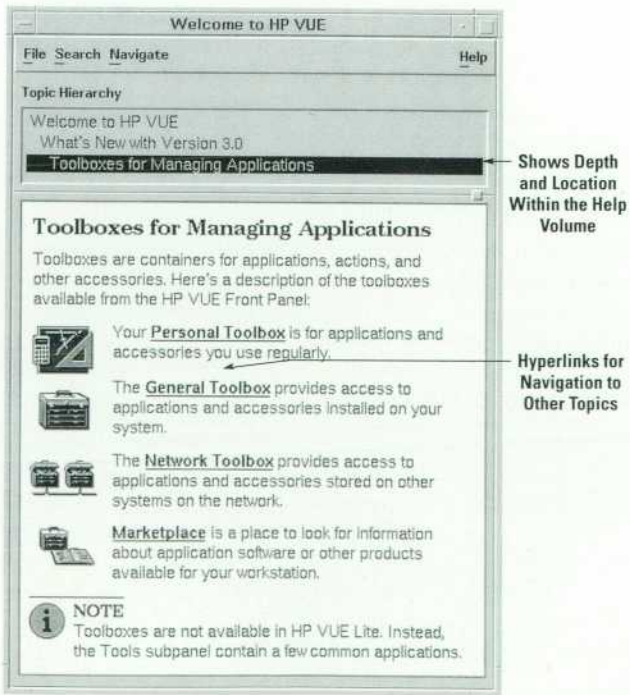


Fig. 2. A sample general help dialog from the HP Help System.

Information Models for Online Help

In designing online help for an application, it is important to organize the information in a way that best fits the application. A tight coupling between the application and its help information typically makes the information more useful when it is needed.

The help topics associated with an application are collectively called a *help volume*. Essentially, a help volume is like a book or document for the application. (However, we avoid the terms "book" and "document" because they tend to skew expectations. Further, some help volumes may have very little in common with their paper counterparts, such as a table of contents, chapters, and so on.)

Typically, there's a single help volume for each application. For very complex applications, or applications with distinct user groups, multiple volumes may be appropriate (Fig. 4).

Standalone Help Volumes. Not all online help is directly associated with an application. Help for the operating system, for example, isn't associated with any single command, program, or utility. Therefore, the information models for online help must allow for standalone help volumes.

Since standalone help volumes are not associated with an application, there must be a way to access them. One method is to provide a help manager. A help manager is simply an application that treats one or more volumes as its own and provides navigation into each volume. A help manager may provide entry points into help volumes that are also accessible from applications as shown in Fig. 5.

Implicit Relationships. Within a help volume, there are two significant organizational models: hierarchical and nonhierarchical. Most technical information is organized hierarchically like a traditional manual, in which chapters contain sections, sections contain subsections, and so on. Relationships between nodes of information are created implicitly based on the hierarchy created by the author. Fig. 6a shows a typical topic hierarchy.

Hierarchical navigation tends to give users more confidence in knowing where they are. There is a sense of depth (number of levels from the top) and movement (up, down, lateral). Hypertext links can be added to connect nonadjacent, but related topics (Fig. 6b). In this case, hypertext is perceived as a bonus, because it provides jumps that span the hierarchical structure.

Explicit Relationships. The free-form nature of a nonhierarchical information web allows authors to organize their information into hierarchies, circles, trails, webs, and grids. Fig. 6c shows an information web. Relationships are created explicitly with hyperlinks. (In some systems, end users can create their own links to connect related topics.)

When navigation is based entirely on hypertext, authors have much more flexibility to create any type of structure. This level of freedom requires a higher level of skill on the author's part. Links to and from each node must be designed and tested carefully.

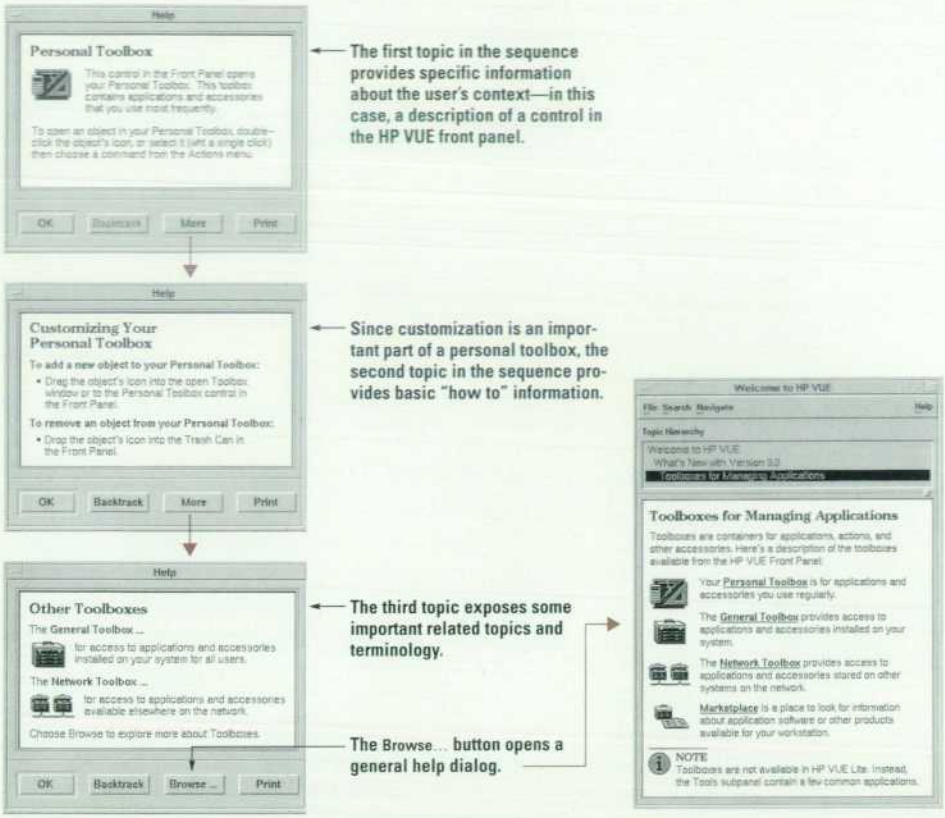


Fig. 3. A progressive disclosure sequence.

Hypertext Links. The lightning speed of hypertext links, and their ability to jump a user to a completely different area of the online information, leaves most users feeling lost after only a few jumps.

System designers and authors must recognize that each hypertext link is an invitation to explore, and therefore, also an invitation to get lost. Most research in this area indicates that hypertext is best used in combination with other navigation aids—"Hypertext is a spice, not a main course."[3]

Usability testing on the HP Help System indicates that when hyperlinks are used to navigate down the topic hierarchy, other links that span the hierarchy should be distinguished in some way. In HP MPower help, most help volumes use a convention of labeling lists of "Subtopics" and "See Also"s to make that distinction.

The HP Help System provides the following features to help users avoid getting lost in hyperspace:

- Backtrack command for undoing a hyperlink and going back to the previous topic
- Topic hierarchy display (in the general help dialog) to show "you are here" information within the current help volume
- History dialog that lists the titles of the topics recently visited
- A "home topic" at the top of each hierarchy to give users a known place to return to if they get lost.

Fig. 4. The association of help volumes in applications. Most help volumes are associated with a simple application.

The Developers' Tasks

There are many facets to developing an application with online help. Depending on organizational structure and the size of the project, these tasks may be performed by a variety of team members.

Planning and Design. Planning and designing a help system for an application requires close collaboration between designers, authors, and programmers. First, the design team must understand the user's need for the application and the tasks that users will perform. From this information a use model can be developed that describes the user's interaction with the application.

As a user interface is designed based on the use model, every effort should be made to keep things as obvious as possible, limiting the need for online help. Horton defines obvious as "... designing products that users already know how to operate ... or can learn by trying things out."[3]

Fig. 5. A help manager provides access to all help volumes.

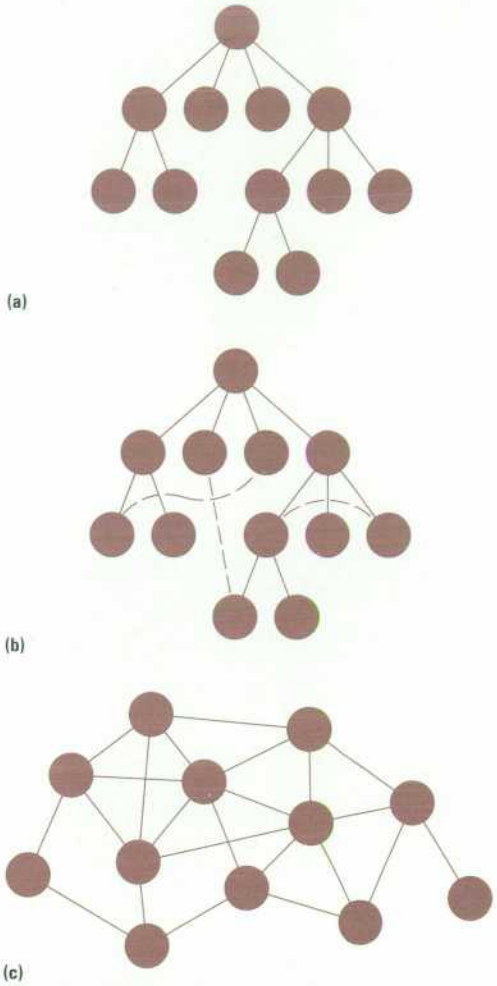


Fig. 6. Help volume organizational models. (a) A typical topic hierarchy. (b) Hypertext links provide shortcuts to span the topic hierarchy. (c) An information web.

For every context within the application that is not completely obvious, identify it as an entry point into help. Assign a unique identifier for connecting the application context to a help topic. Most help systems have an identifier naming scheme. In the HP Help System an identifier can be any string up to 64 characters including letters, digits, and hyphens.

Authoring. The author's job includes writing all of the help topics, assigning the corresponding identifiers along the way. For each help topic, the author must consider the many ways the topic may be read. For example, a topic that is typically seen as an entry point (displayed directly when the user requests help) may also be part of the browsable topic hierarchy. The topic must be authored to work well in both contexts.

The breadth and depth of exposure that an author has to the product during the writing process provides an opportunity to discover design flaws that may decrease usability. It continues to be the author's responsibility to reduce the need for online help by exposing these kinds of problems.

Programming. The software developer is responsible for providing the hooks from the software into online help. This requires using the agreed upon topic identifiers for displaying topics when the user requests help.

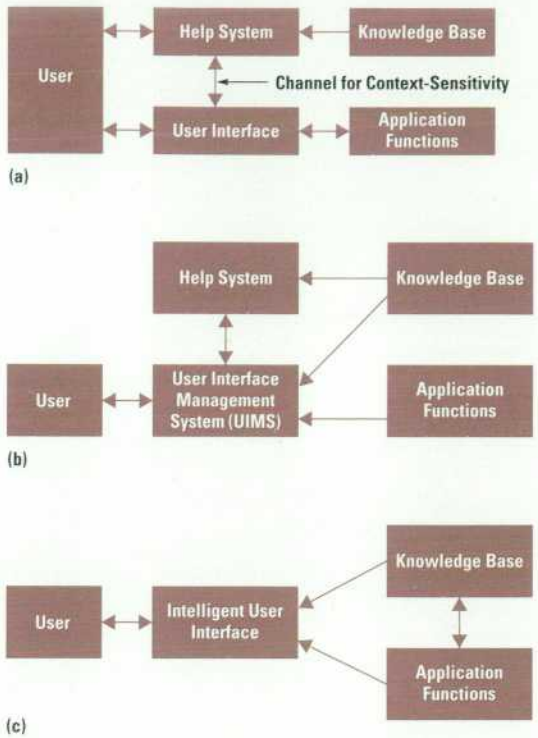


Fig. 7. Application help architectures. (a) The traditional application help architecture. (b) A possible migration of the traditional application help architecture. (c) Another possible migration of the traditional application help architecture.

Depending on the help system being used, there may be some additional programming necessary to achieve the desired behavior. For instance, with the HP Help System, progressive disclosure sequences like the one shown in Fig. 3 require the application to provide the correct behavior for the More button that walks through the sequence.

The Future of Online Help

Much of this paper has assumed a traditional approach to online help development. That is, one in which the help system and its supporting information are aligned with the application's user interface and supporting functions. Context sensitivity is enabled by direct connections between the help system and user interface components and states within the application. Fig. 7a shows a traditional application help architecture.

Reference 4 describes a help system whose design is driven by the same knowledge base that drives the user interface (via a user interface management system, or UIMS). In this model, the flow of information and functionality is all handled through the UIMS which presents the user interface for the application and the help system (see Fig. 7b).

Online information systems of the future may evolve from architectures like the one suggested by Foley and colleagues.[4] However, I expect help systems to meld further into the application, to the point that the help system itself isn't identified as a discrete component.

As this takes place, the engineering process for the information and knowledge portions of products will be as important as the traditional hardware and software engineering tasks. In

fact, the information architecture will become increasingly difficult to distinguish from the application architecture.

An intelligent user interface will draw upon an information base and the application functions to present user interface components (see Fig. 7c). The user interface may still present what the user perceives as online help, but it is generated from a combination of knowledge and application functions, along with the user's current context. Further, there will be a strong relationship between the knowledge base and the application functions (a channel missing in Foley's model).

Conclusion

Providing online help has grown in recent years from a "nice to have" feature to an expected component of all significant application software. For some products, online help reduces support costs because it puts information where the user is more likely to find it (meeting the goals in our goal hierarchy). For other products, providing online help also reduces the cost of printing hardcopy manuals.

As applications continue to grow in complexity, so does the need to simplify them for users. For the foreseeable future, online help plays an important role in helping with that simplification. Today, online help systems are discrete components that aid application developers in packaging easier-to-use products. In the future, online help will become a by-product of an information engineering process that addresses all information aspects of a product.

Acknowledgments

Gathering and refining of the information presented in this paper wouldn't have been possible without the contributions of many people throughout HP during the HP Help System project.

Richard Artz, Anna Ellendman, and Jodi Peterson were among the dozens of talented learning products engineers that contributed valuable design review and direction for online learning products. Jay Lundell and Jan Ryles offered insightful human factors review and testing.

Of course, the engineers who designed and built the HP Help System contributed the most important part—a great product. Thanks to Lori Cook, Brian Cripe, Axel Deininger, Steve Hiebert, and Mike Wilson. Also, thanks to Bill Kaster, loaned to us from Corporate Learning Products, for codeveloping the HP HelpTag language and compiler. Project managers Bob Merritt, Bob Miller, and Rick McKay also helped to foster a creative environment by welcoming cross-functional and cross-organizational collaboration.

References

1. N. J. Belkin, and P.G. Marchetti, "Determining the Functionality and Features of an Intelligent Interface to an Information Retrieval System," *Proceedings of the 13th International Conference on Research and Development in Information Retrieval* (SIGIR '90), September 1990, pp. 151-178.
2. J. Price, "Creating a Style for Inline Help," *ConText and HyperText—Writing With and For the Computer*, MIT Press, pp. 311-323.
3. W. Horton, "Let's Do Away With Manuals Before They Do Away With Us," *Journal of the Society of Technical Communications*, Vol. 40, no. 1, January 1993, pp. 26-34.
4. J. Foley, W. Kim, S. Kovacevic, and K. Murray, "UIDE – An Intelligent User Interface Design Environment," *Intelligent User Interfaces*, Addison-Wesley Publishing Company.

Authors

April 1994

6 Development of a Multimedia Product

Gary P. Rose

Gary Rose is a native of Springfield, Massachusetts and attended the University of Connecticut, where he received a BS degree in computer science and electrical engineering in 1981. He worked at Digital Equipment Corporation, then moved to Apollo Computer and joined HP when Apollo was acquired in 1989. At Apollo, he managed the development of a software graphics kernel and designed other graphics software as well. Now an R&D systems programmer at the Workstation Technology Division, he is the multimedia program manager responsible for HP MPower. Gary is married and has two sons.

Jeffery T. Oesterle

Software engineer Jeff Oesterle received a BS degree in computer engineering from Carnegie-Mellon University in 1989 and joined HP's Graphics Technology Division the same year. He developed graphics hardware for two HP workstation product series and worked on the design, architecture, and software engineering teams for HP TeleShare. He is currently at the Workstation Technology Division. Jeff's outside interests include T'ai Chi Ch'uan, skiing, and transpersonal psychology.

Joseph E. Kasper

Joe Kasper was born in New Haven, Connecticut, and attended Saint Michael's College, from which he received a BA degree in English and education in 1970. He continued his studies at Lesley College, receiving a master's degree in education in 1978. He worked as a technical writer at Computervision for several years before moving to Apollo Computer and joining HP in 1989 when Apollo was acquired. He wrote documentation for graphics programming and is now a learning products engineer at the Workstation Technology Division, where he writes programming documentation for multimedia products. He wrote the online help and tutorial information as well as the programmer's manual for HP MPower's digital video. He's also the coauthor of a book on graphics programming. Joe and his wife have two children. Outside work, he plays piano and organ, and enjoys hiking and fishing.

Robert J. Hammond

An R&D project manager at HP's Workstation Technology Division, Bob Hammond is responsible for the digital video product offered in HP MPower. Bob has been with HP since 1989, when Apollo Computer was acquired. On earlier HP projects, he was a software engineer, project engineer, and then a project manager for graphics and audio technology. Before joining Apollo he designed phototypesetting software at Compugraphic Corporation. He's the author of two technical papers. Bob was born in Allentown, Pennsylvania and attended the University of Connecticut, from which he received a BS degree in biology in 1980. He's married, has three children, and likes scuba diving, basketball, and ice hockey.

10 HP MPower

William R. Yoder

Born in Bloomington, Illinois, Bill Yoder taught high school and college English before joining HP in 1980. Now a technical contributor at the Workstation Technology Division, he started work at HP as a senior technical writer and has contributed to the development of several HP products, including HP VUE 2.0 and 3.0. He was a project leader on the HP MPower project. He received an AB degree in history and literature from Harvard University in 1972 and an MA degree in English from the University of California at Santa Barbara in 1976. Following his change in careers, he studied computer science, completing work for a BS degree in 1985 from Oregon State University and for an MS degree in 1988 from Stanford University. He is the author of several conference papers and one previous HP Journal article, and is a member of the ACM, the IEEE, and Computer Professionals for Social Responsibility. Bill is married, has a son, and works on a local political committee. Outside work, he enjoys mountain biking, jogging, reading, and playing in a rock-and-roll band.

20 Graphical User Interface

Charles V. Fernandez

Charlie Fernandez joined HP in 1988 and is a product manager for multimedia software at the Workstation Technology Division. He was a product manager for HP VUE 1.0 and 2.0, HP TeleShare, HP MPower's digital video, and the HP MPower Developer's Kit. Earlier, he was a technical writer for documentation describing how to use the X Window System and beginner's guides for OSF/Motif and HP VUE 1.0. Before coming to HP he was a freelance

writing consultant and worked at several startup software companies. Born in Albany, New York, he studied English at the University of Detroit (BA 1972) and at the University of Oregon (MA 1975). He's the author of two previous HP Journal articles and is a member of the Society for Technical Communication. Charlie is married. An avid fly-fisherman, specializing in large, native rainbow trout, he has written about the topic and is a member of three fishing organizations. He also enjoys carpentry.

23 HP SharedX

Daniel Garfinkel

A software engineer at HP's Advanced Systems Division, Dan Garfinkel joined the Corvallis Division in 1981. On past projects, he designed and implemented a user interface and a software calculator for the HP Integral personal computer and was also a project leader for the first implementation of the X Window System on the HP-UX operating system. At HP Laboratories for three years, his research interests led to an architecture for high-speed graphics in the X Window System and a prototype of the HP SharedX system. Transferring to Fort Collins with the HP SharedX prototype, he was the project leader for HP SharedX and was the architect and implementer for the HP SharedX Extension. In addition, he designed and implemented the user interface for the HP SharedX Whiteboard. His work has resulted in a patent on three-dimensional memory management and three pending patents related to HP SharedX. He coauthored a 1991 conference paper on application-sharing architectures and reviews groupware articles for IEEE Computer Magazine. A native of East Lyme, Connecticut, Dan received a BS degree in computer science in 1981 from the University of Connecticut. He's married, has two daughters, and likes basketball, softball, sailing, and travel.

Bruce C. Welti

Software design engineer Bruce Welti was born in Seattle, Washington and attended Antioch University, from which he received a BA degree in psychology and computer science in 1988. He joined HP the same year, initially working at the Electronic Design Division and later transferring to the Network and System Management Division. He worked on a printed circuit board autorouter and developed media management software for a magneto-optical disk autochanger. For HP SharedX, he developed automatic font matching, color matching, multivisual screen capture algorithms, and the command-line interface for HP SharedX. Currently, he's designing and implementing "user roles" in HP OpenView. He is named as an inventor on two patents on font and color matching. Bruce is married and has three children. His outside interests include classical violin, Aikido, and flying airplanes.

Thomas W. Yip

With HP since 1992, Tom Yip is a software design engineer at the Advanced Systems Division. His contributions on HP SharedX include development and internationalization of the HP SharedX Whiteboard as well as product enhancement and performance tuning. His work on the HP SharedX project has resulted in a patent pending on color matching for computer displays. He's currently working on a Microsoft Windows emulator for the HP-UX operating system. A native of Newport News, Virginia, his BS degree in computer engineering was awarded in 1992 by Virginia Polytechnic Institute and State University. During his college years he performed atmospheric research at the NASA Langley Research Center. Tom enjoys racquetball, mountain biking, and skiing.

37 Imaging Services in a Multimedia Environment

Andrew F. Munro

A learning products engineer at the Workstation Technology Division, Andy Munro joined HP in 1989. He was born in Glasgow, Scotland and received a BS degree in video and film production from Emerson College in 1976. He also has a master's degree in instructional media design from Boston University (1981) and has completed graduate-level courses in technical communication at the University of Lowell. His work on the HP Image Library includes writing a programmer's manual and online help and creating installation scripts for HP ImageView. He is currently developing installation and testing scripts and writing online help and printed documentation for HP MPower and HP SharedPrint. Earlier, he developed documentation for FDDI (fiber distributed data interface) and other networking products. Before coming to HP he worked at Prime Computer, where he was a documentation project leader and writer. His professional interest is developing intelligent scripts that make software easy to use. Andy is married and has a daughter. His outside interests include acting in community theater, playing piano, and photography.

Ahmad H. Shekarabi

Ahmad Shekarabi has been with HP since 1989, when the company acquired Apollo Computer, and is currently a member of the technical staff at the Workstation Technology Division. He has worked on circuit and chip design for memory subsystems and graphics hardware for several Apollo products and more recently contributed to the development of the HP Image Library and HP ImageView. Before joining Apollo he worked at Computervision Corporation. Ahmad was born in Tehran, Iran. He completed work for a BSEE degree from Norwich University in 1980 and has done graduate-level work at the University of Vermont and at Lowell University. He is married and has three children. In addition to spending time with his family, he enjoys skiing, hiking, and running.

44 Printing Solution for a Multimedia Environment

John Mandler

Software designer John Mandler was born in Brooklyn, New York, received a BSEE degree from Northeastern University in 1973, and completed a masters program in biomedical engineering from the University of Connecticut in 1978. He has been with HP since 1989, when Apollo Computer was acquired, and is now working at the Medical Products Group where he develops and implements software for patient monitoring systems. While at the Apollo Division, he worked on the design, implementation and support of Domain/OS hardcopy subsystems and I/O controllers. He also was the lead product engineer for HP SharedPrint. His other professional experience includes working on a phased-array ultrasonic imaging system and a fetal heart monitor. He's a member of the IEEE. John is married and has two daughters. He likes woodworking and outdoor activities such as cross-country skiing, hiking, and gardening.

53 Faxing Documents in HP MPower

Francis P. Sung

A software engineer at the Workstation Technology Division, Francis Sung joined HP in 1989, when Apollo Computer was acquired. He developed portions of the software for the HP MPower fax utility and is currently working on support for digital video. Earlier, he contributed to the development of a graphics display library and digital audio support. He worked at Compugraphic Corporation and at Data General before joining Apollo. He is a member of the ACM and coauthor of two technical articles. Francis was born in Hong Kong and studied electrical engineering at Brown University, from which he received a BSEE degree in 1979 and an MSEE degree in 1981. He's married and has four children. His outside interests include fixing cars and other things around the house, church activities, classical music, photography, singing, and cooking.

Mark A. Johnson

Mark Johnson was born in Point Pleasant, New Jersey and educated at the University of Maine, from which he received a BSEE degree in 1981. He designed graphics hardware for CAD/CAM workstations at Computervision Corporation, then joined Apollo Computer, where he designed a gate array for an Apollo graphics workstation. He joined HP in 1989 when Apollo was acquired. He was a member of the original development team for the HP 9000 Series 700 controllers, and worked on the architecture, design, and implementation of the graphical user interface for the HP MPower fax utility. He's currently responsible for developing the fax software component for the next version of HP MPower at the Workstation Technology Division and is a member of the IEEE. Mark is married and has two daughters. He's active as a bicycle road racer, enjoys spending time with his daughters and likes mountain biking, cross-country skiing, and brewing beer.

62 Audio Support in HP MPower

Ellen Nordahl Brandt

A member of the technical staff at the Workstation Technology Division, Ellen Brandt received a BSEE degree from Cornell University in 1983. She worked on graphics hardware at Computervision, then went to Apollo Computer and joined HP in 1989, when Apollo was acquired. She developed graphics hardware for Apollo products and worked on the audio library for HP MPower. Ellen is married and has one son.

Thomas G. Fincher

Tom Fincher graduated in 1980 from Stanford University with a BSEE degree and worked at Wang Laboratories and CalComp before going to Apollo Computer. He joined HP in 1989 at the time of Apollo's acquisition. His background includes developing graphics hardware for Apollo products and audio GUIs for HP MPower. He is currently a member of the technical staff at HP's Workstation Technology Division.

Monish S. Shah

Monish Shah is a graduate of Texas A&M University (BSEE 1984) and Stanford University (MSEE 1989) and an engineer/scientist at HP's Advanced Systems Division. He worked on several graphics products, including HP SRX, HP Turbo SRX, and built-in graphics for HP 9000 Models 705 and 710 workstations. He designed the interface ASIC for the audio hardware for HP MPower and is now developing graphics products and hardware for HP workstations. His work has resulted in three patents related to graphics.

68 Video Support in HP MPower

Craig S. Richard

Craig Richard is a member of the technical staff at HP's Workstation Technology Division and has been with HP since 1989, when Apollo Computer was acquired. His specialty is multimedia technology, particularly digital video. He has a BSEE degree (1984) from the University of Massachusetts and developed graphics software at a startup company and at CalComp before joining Apollo. At Apollo, he worked on advanced graphics features such as multiple color maps and texture mapping. He currently develops and manages relationships with independent hardware and software vendors, integrating some of their technology into HP multimedia products. Craig was born in Melrose, Massachusetts, is married, and has two sons. In addition to spending time with his family, he likes boating, hockey, softball, skiing, and scuba diving.

71 Mail Facilities in a Multimedia Environment

Robert B. Williams

A software engineer at the Workstation Technology Division, Robert Williams has been with HP since 1982. He was a member of the development team for HP VUE and later worked on the HP MPower mail facility and on HP MPower system integration. He is currently migrating HP VUE to the HP-UX 10.0 operating system and working on distributed computing issues. A member of the ACM, he's especially interested in user interface design in networked environments. Robert was born in Wisconsin and completed work for a BS degree in computer science from Oregon State University in 1988. His favorite leisure activity is sailing; he races and teaches a sailing class. He also enjoys mountain biking and gardening.

Harry K. Phinney

With HP since 1979, Harry Phinney is a software design engineer at the Workstation Technology Division. In the past he provided customer support for calculators and technical support for desktop computers. He helped develop HP's first X Window System server and was the lead engineer for HP's first X11 system server. More recently, he designed and developed software for the HP MPower mail facility and text editor. Harry was born in Corvallis, Oregon and has a BS degree in mechanical engineering from Oregon State University (1979). He's married and enjoys bicycle racing and restoring sports cars.

Kenneth L. Steege

R&D software engineer Ken Steege has been with HP since 1982 when he joined the Corvallis Division. Initially, he was a support engineer for the R&D calculator lab and later developed connection management software for the HP Portable Plus. He also developed X11 client applications for the HP-UX operating system. Now a member of the Workstation Technology Division, he contributed to the development of the HP MPower mail facility and HP MPower integration. He's currently responsible for text editors for HP-UX 10.0. Ken was born in Denver, Colorado and attended the University of Oregon, from which he received a BS degree in computer science in 1972. He spent 10 years as a programmer/analyst and consultant on small IBM Corp., Digital Equipment Corporation, and HP business systems before coming to HP. He is married and has three children and four grandchildren. He participates in a prison fellowship program and is involved in home schooling. His other interests include fishing and gentleman farming.

79 Fast and Intuitive Online Help System

Michael R. Wilson

With HP since 1987, Mike Wilson is a member of the technical staff at the Workstation Technology Division. Mike played an integral part in the development of the HP Help System through the releases of HP VUE 2.0 and 3.0. Mike's specific responsibilities included overall system design, public application program interface specifications, user interface design and development, and custom widget creation. He's now working on the next version of the HP Help System. Mike was born in Palo Alto, California and attended California State University at Chico, from which he received a BS degree in computer science in 1987. He is married and has two sons. He and his family enjoy bicycling.

Lori A. Cook

Lori Cook is a member of the technical staff at the Workstation Technology Division. She joined HP in 1980, the same year she completed work for her BS degree in computer science from Oregon State University. She has worked on the mass storage, I/O, and service ROMs for the HP 87 personal computer, wrote the electronic disk ROM for the HP 85 and HP 87 personal computers, and worked on an HP-IB driver and an HP-IB command library for HP personal computers. She is currently responsible for reading and rendering the text and graphics displayed in help dialogs for the HP Help Developer's Kit. Lori was born in Tacoma, Washington. She's married and her hobbies include scuba diving, skiing, reading, knitting, and racquetball. Her newest interest is rock hunting.

Steven P. Hiebert

Steve Hiebert was born in Portland, Oregon. He served in the U.S. Air Force for four years, attaining the rank of staff sergeant. He attended Portland State University from which he received a BS degree in mathematics in 1976. He was a software engineer at Electro Scientific Industries and manager of software engineering at TimeShare Corporation before joining HP in 1981. He's currently a member of the technical staff at the Workstation Technology Division. He worked on compilers, optimizers and loaders for the HP Integral PC. He later developed shared libraries and compression techniques for the executable and data files that went on the software engineering ROM for the HP Integral PC. He has also done development work on servers for the X Window System, version 11 and on the HP Interface Architect. His work on the HP Help System included internationalization and compilation of the HP HelpTag files into an internal distribution format. He is now working on the next version of the HP Help System. Steve is a member of the IEEE and ACM and is an author of a previous HP Journal article on merging Starbase and X11.

90 Developing Online Application Help

Dex Smith

Dex Smith is a graduate of Oregon State University and has a BS degree in technical journalism with a computer engineering minor (1987). Before joining HP in 1988, he was a freelance writer, working mostly on HP calculator manuals. As a learning products engineer for the Workstation Group's user interface laboratory in Corvallis, Oregon, he wrote the manuals for a variety of user interface products, including X11, OSF/Motif, and HP Interface Architect. He was the lead learning products engineer for the HP VUE 3.0 online help system, as well as a contributor to the requirements assessment and design of the HP Help System. Dex is currently an engineer/scientist working for HP Corporate Engineering as a company-wide consultant, serving all of HP's product groups in the areas of information engineering, user interface design, and multimedia technology. He is a member of the ACM. Dex and his wife have two sons.

Fr: Worldwide Roster/190LDC 00107389 *
5731

To: LEWIS, KAREN
HP CORPORATE HEADQUARTERS
DDIV 0000 20BR
IDR #20088

HEWLETT-PACKARD
JOURNAL

April 1994 Volume 45 • Number 2

Technical Information from the Laboratories of
Hewlett-Packard Company

Hewlett-Packard Company, P.O. Box 51827
Palo Alto, California, 94303-0724 U.S.A.

HEWLETT
PACKARD

5962-7972E