InternetExpress: An Inter-Desktop Multimedia Data-Transfer Service

Murugappan Palaniappan and George Fitzmaurice, Brown University*

Featuring a simple desktop operation that works uniformly across all applications, the InternetExpress integrated service enables users to send and receive selection-based multimedia data in a timely manner.

Internet users today need to share multimedia data like text graphics, spreadsheets, and images as much as they need to share ASCII text.[1] Currently, multimedia documents are often transmitted in printed form via fax and express mail. However, sharing electronically stored data in electronic form is more efficient than in printed form. Accessing and modifying an electronic version is faster and more convenient than sharing and annotating a printed copy.[2]

Consider the following scenario:

> Joe and Mary are co-designing the schematic layout of a new computer chip. They work at different sites and use different but compatible CAD/CAM software packages. Joe works on an IBM PC using the L-Edit package in San Jose, California, while Mary works on a Sun workstation using the Magic package in Washington, D.C. Joe finishes one component of the design and transfers it to Mary to check for consistency with her design. A single command from within the application completes the transfer transaction. At the time of Joe's transmission, Mary has already left her office for the day. When she arrives at work the next morning, she sees that she has received an electronic express package from Joe. When she opens it, Joe's component is presented to her in Magic format. She finds something in Joe's design that does not match their proposed specifications. She modifies the design accordingly and telephones Joe to discuss her proposed changes. As their conversation starts, she selects the modified component and sends it to Joe. In a few seconds, a message on his monitor indicates that Mary's package has arrived. He views it immediately in L-Edit format, agrees with the changes, updates his copy of the document, and resends it to Mary.

This scenario illustrates many of the features needed in a data-transfer service. Users who work with compatible applications, possibly across heterogeneous platforms, should be able to share data easily and quickly. Regardless of the application, users should be able to send data by issuing a single command. They often need to send data at the selection level rather than at the document level. The responsibility for determining a compatible application for presenting the data should fall on the service and not on the user.

To date, sharing ASCII text has been achieved primarily by electronic mail. This

* Since collaborating on this article, Palaniappan joined Aldus Corp. and Fitzmaurice started doctoral studies at the University of Toronto.

0018-9162/91/1000-0058$01.00 © 1991 IEEE

tool is quite independent of any application and often requires the author to directly enter the ASCII text using a simple editor before sending. Indeed, the success of electronic mail can be attributed to its widespread availability across heterogeneous environments, its simple interface, its reliability, and its low cost.

Sharing multimedia data, however, is inherently more difficult than sharing ASCII text. Unlike ASCII text, multimedia data is created within an application (for example, pictures are most likely created within a graphics application). In current practice, these applications have not been designed to transport the data to users at different sites. Instead, network-based tools, such as file-transfer protocol (FTP), and phone-based tools, such as facsimile machines, have been used for transferring the data.

Most existing tools fall short in one or more of the following areas:

- *heterogeneity* — the ability to work across different hardware and software platforms;
- *seamlessness* — the ability to work uniformly across all applications;
- *user-interface simplicity* — ease of use, requiring few steps for a single transfer;
- *timeliness* — the ability to transfer data almost instantaneously; and
- *security* — preventing outside users from accessing restricted data.

To overcome the shortcomings in existing network-based transfer tools, we have designed an integrated desktop data-transfer service called InternetExpress. We chose the name InternetExpress to denote quick, reliable, and timely transfer of multimedia data on the Internet. This service allows users to share data conforming to any data standard across heterogeneous environments on the Internet, assuming the environments follow the general principles of the desktop metaphor.[3,4] InternetExpress is selection-based and works seamlessly across all desktop applications. A prototype of InternetExpress has been implemented in the Intermedia desktop environment.[5]

Background

Before describing the InternetExpress service, we critique numerous solutions currently used for transferring multimedia data. Then, we summarize the characteristics that make these solutions successful.

InternetExpress lets users share data conforming to any data standard across heterogeneous environments on the Internet.

Related network-based solutions. Many network-based tools such as FTP and Kermit have been developed to transfer data across wide area networks. But we believe these tools fall short in that they impose burdens on the users and raise security concerns.

A user usually cannot deposit documents in another user's directory at a remote site unless the user has access rights to connect. Consequently, a sender initiates the transfer by asking a recipient to retrieve the document at a specified location. The sender must also set up permissions so that the recipient can connect to the sender's system to retrieve the document.

To set up permissions, the sender could disclose his or her user password or get the system administrator to set up a special, restrictive account for the recipient. Once connected, a recipient is usually able to browse other documents on the system and has the potential ability to modify or deposit documents.

After retrieving the document, the recipient still might not be able to interpret the data easily. For instance, a sender could save an Excel spreadsheet document in an industry data standard such as WK1 and inform the recipient of this. If the recipient does not have an Excel application, he or she has to know that it is still possible to view the data using a Lotus 1-2-3 application that reads the WK1 file format.

A second approach to sharing documents is to rely on a remote file system mounting capability, as in the Andrew File System.[6] Like the previous approach, this solution places many burdens on users and raises security issues. By sending information about the location of a document (via electronic mail, for example), a sender can ask a recipient to remotely mount the file system on which the document resides and either access or make a copy of the document.

For a file system to be remotely mounted, the sender must get the system administrator to grant permissions to the relevant sites. The system administrator has to update one or more system files, both when granting and when removing access privileges. At the recipient's site, the system administrator must mount and unmount the file system, as necessary. Once the file system is mounted, a recipient can access any documents on the file system as though they are on a local file system. Again, a recipient may not be able to easily interpret the accessed or retrieved data.

In both approaches, the sender is more likely to initiate the transfer transaction. This can present a problem if a receiver does not retrieve the data immediately. For example, the document could be moved, renamed, or modified.

In another approach, a user on a Unix-based machine can send multimedia data as an electronic mail message. This solution requires the sender to convert any binary multimedia data into ASCII form using the *uuencode* command. Once in ASCII form, the data can be included as part of the message. The recipient then has to extract the data from the message and/or restore it as binary multimedia data using the *uudecode* command. This approach is cumbersome and involves multiple steps.

A fourth approach developed recently by several computer vendors lets users transfer multimedia documents conforming to the data formats within their specific packages. For example, BBN/Slate, IBM/Office Vision, HP/NewWave Office, and DEC/All-in-One are similar packages that contain a suite of applications for creating and sharing multimedia documents. In most of the packages, the applications display only the data, while relying on electronic mail to physically transfer the data.

These self-contained packages let users transfer multimedia data only to sites that have the specific package installed; each package imposes restrictions on adding new data types. A recipient who does not have the package can obtain an ASCII text version of the multimedia documents. For example, an image would be represented as one

line of text stating that the picture would have appeared at that location. The primary limitation of this approach is that the transfer mechanism is part of a self-contained package and is not fully integrated into each vendor's desktop.

A fifth approach is to use vendor-specific, enhanced electronic mail systems, such as Microsoft/MS Mail and CE Software/QuickMail. Both systems follow the Macintosh-style graphical user interface and are easy to use. While sending an electronic mail message, QuickMail users can include a selection previously placed on the clipboard. When the recipient reads the message, the accompanying multimedia data is placed on the recipient's clipboard. Using a clipboard, however, is problematic, since it prevents a user from sending multimedia data to people who do not have clipboards, and it destroys the resident contents of the recipient's clipboard.

Both systems are designed to operate on local area networks and need modems to communicate with other sites, thereby restricting data-transfer timeliness. In addition, these services require the recipient to know the appropriate application to use to interpret the incoming data.

Although it is only an ASCII data-transfer mechanism, the Unix *talk* program is a sixth approach that is architecturally interesting. This utility allows users to communicate across heterogeneous machines on the Internet by sharing data in real time. A user can initiate the communication in a single step and be notified of the presence or absence of other parties.

This program uses point-to-point connections as opposed to the store-and-forward approach used by electronic mail. Users, however, can only communicate with others who are active at their workstations or terminals. Further, the utility uses an abrasive notification mechanism to inform other parties of the communication request.

Related technologies. Facsimile machines[7] transfer bitmap images of multimedia data over telephone lines. The image transmission takes place over a high-bandwidth, point-to-point connection. These dedicated fax machines have been successful because they offer fast communication, use the existing telephone-line infrastructure, have a simple user interface, and can send any handwritten or printed material. But, since a fax is sent to a machine, usually a shared resource, there is often a delay between arrival at a machine and receipt by the individual.

One improvement to using a fax machine is to add a fax modem to a computer. The fax modem automates document transmission directly from the computer, eliminating the need to print a paper copy. Examples of fax modems include Apple's AppleFax, Abaton's InterFax, and STF Technologies' FaxSTF. These modems let you store faxes in a proprietary format to be sent to a fax machine or a computer equipped with a fax modem, but they do not support industry data standards or provide imaging or manipulating capabilities.[8] Extending the AppleFax software with Solution's BackFax software allows background transfer of not only faxes but also binary and text files. The files, however, can only be transferred to sites that have an AppleFax modem and BackFax software.

Vendor-specific networks such as AppleLink can be considered another approach to document transfer. AppleLink consists of a modem-accessed telephone network and an application to create, send, and receive documents. The AppleLink network is connected to a mainframe that deals with document routing and storage. Registered users can send ASCII text messages or Macintosh files to AppleLink users. This approach works well among Apple computers but can't work across heterogeneous systems.

Solutions International's Super Glue II provides a partial solution to the problem of sharing multimedia data by defining a standard Glue format and supplying a system-viewer application. The utility captures the "printed output" of a document on disk (in Glue format) instead. This Glue file can then be sent using standard transport mechanisms such as AppleLink. The recipient of a Glue file can view the data with the system-viewer application. This utility does not preserve the original format of the data, thereby preventing a recipient from manipulating the data. Converting the data to Glue format is a multistep procedure.

Although it does not directly address the problem of sharing multimedia data, Software Innovation markets an interesting Macintosh file-launching utility called HandOff II, which allows users to work with foreign file formats. In the Macintosh environment, double-clicking on a data file brings up the application in which the data was created. If the application does not exist, an error message is presented; the user must launch a compatible application that can interpret the data.

The HandOff II utility provides a mechanism for users to bypass this procedure. For example, a user may not be able to open a file created by a Lotus 1-2-3 application. The utility, however, would recognize the WKS format and present the data in a compatible application like Excel.

Another popular method of sharing multimedia data is to use overnight delivery services such as Federal Express. Since this method is based on postal addresses, documents can be transferred reliably to almost anyone. This service burdens the user, who has to print and package the document, fill out the necessary forms, and abide by rigid drop-off times for sending and receiving documents.

Important characteristics of data-transfer mechanisms. After surveying the data-transfer mechanisms, we recognized characteristics that made each approach successful. We then identified 12 important characteristics that we believe constitute a complete solution to transferring multimedia data. Table 1 summarizes how each system holds up to these characteristics. Each of the surveyed systems falls short in two or more categories. Most notably, the data-transfer mechanisms often have cumbersome interfaces, lack software and hardware heterogeneity, and separate the transport mechanism from the applications.

We also recognize that cost is an important factor in determining the frequency of use of a data-transfer system. Comparing the cost of transferring data across different technologies, however, is beyond the scope of this article. In general, network-based technology is relatively cheaper than phone-based tools and overnight delivery services. Users can afford to transfer data frequently over the Internet domain because it is currently free for most users; the government covers the cost involved in both transferring the data and maintaining the network.

We developed InternetExpress by considering the strengths of some of the

Table 1. Characteristics of data-transfer mechanisms.

Name of System	Fine Granularity	Seamlessness	Hardware Heterogeneity	Software Heterogeneity	Timeliness	Single Step	Preserve Format	Notification	Confirmation	Enclosed Message	Security	Wide area Network
FTP/Kermit			Y				Y					Y
Andrew File System		NA	Y	NA	Y	Y(1)	Y					Y
Unix E-mail		NA	Y	NA			Y	Y		Y	Y	Y
BBN Slate			Y				Y(6)	Y		Y	Y	Y
Office Vision			Y(2)				Y	Y	Y	Y(7)	Y	Y
NewWave Office							Y	Y(8)	Y(8)	Y	Y	Y
DEC All-in-one			Y				Y	Y	Y	Y	Y	Y
QuickMail	Y	Y(3)			Y(4)		Y	Y	Y	Y	Y	Y(9)
Microsoft Mail		Y(14)	Y		Y(4)		Y	Y	Y	Y	Y	Y(9)
Unix *talk*	NA	NA	NA	NA	Y	Y	NA	Y	Y	NA	Y	Y
Fax		Y(5)	NA	NA	Y					Y	Y(10)	Y
Glue with Apple Link	Y	Y	Y	Y				Y			Y	Y(9)
Apple Link							Y	Y	Y		Y	Y(11)
Federal Express			NA	NA			Y(12)	Y	Y	Y	NA	Y(13)
Internet Express	Y	Y(14)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Definitions

Fine granularity:	Can data at the selection level be transferred?
Seamlessness:	Can all desktop applications participate in the systemwide service?
Hardware heterogeneity:	Is the data transferrable across different hardware platforms?
Software heterogeneity:	Can the desktop automatically launch applications based on data type?
Timeliness:	Is the data transferred directly from sender to receiver in a single step?
Single step:	Does it take only one command for the user to send or receive data?
Preserve format:	Does the transfer maintain the original electronic format of the data?
Notification:	Is the user informed when data arrives?
Confirmation:	Can a sender ask to be informed when a recipient receives the data?
Enclosed message:	Can a message be enclosed along with the data being transferred?
Security:	Is the receiver prevented from browsing/modifying documents in the sender's file system?
Wide area network:	Can data be transferred to any site on the Internet?

Notes

(1) Initially, requires many steps to connect. (2) Works only across IBM PC compatibles. (3) Uses a system clipboard for the transfer. (4) Achieved by using a modem. May be a problem if modem is already in use. (5) Fax boards allow applications to participate uniformly by using the print mechanism. (6) Writes and transfers only vendor-defined multimedia data. (7) Fixed-length message. (8) Depends on underlying electronic mail service. (9) Needs a modem to connect to other local area AppleTalk networks. (10) An issue at the receiver's end, since one user may be able to pick up another user's fax. (11) If AppleLink has access to Internet. (12) Only physical entities can be preserved. (13) Uses postal addressing scheme. (14)If the applications comply with the APIs.

more prominent data-transfer mechanisms. We wanted a mechanism akin to Unix electronic mail that would transfer multimedia data over heterogeneous systems across the Internet. Unlike electronic mail's store-and-forward technique, we opted for a direct, single-step point-to-point transfer method, analogous to fax machines and Unix *talk*, to ensure fast and reliable delivery. Our approach, unlike Unix *talk*, does not require recipients to be available or logged on for the transfer to occur. We borrowed the cover-sheet concept from fax machines to allow users to enclose text messages and specify one or more recipients.

Like MS Mail, we defined an application programmer's interface to extract data from applications. We further extended the API to present incoming

data in an appropriate application, a process similar to the functionality of HandOff II. Also, we believe users want to share data at the selection level, as in QuickMail.

Following the conventions of several data-transfer mechanisms, we provide notification and confirmation in our solution. InternetExpress borrows the notion of packages used by overnight delivery services and provides notification by using an express-package icon that sits on the user's desktop. Like OfficeVision and All-in-One, our system lets users track the status of a sent message.

Overview

With our architecture, selections of any type of data can be sent across the Internet and received within seconds by any machine that implements the InternetExpress service. Since our service is based on sending, not retrieving, security is not an issue. Because InternetExpress operates on the Internet, data transfer is as inexpensive as electronic mail.

We have implemented an InternetExpress prototype in the Intermedia desktop environment and tested it across different local area networks at Brown University. Intermedia applications that support this data-transfer service include InterDraw, a structured graphics editor; InterWord, a word processor; and InterVal, a timeline editor. InterDraw and InterVal write their data in PICT format, while InterWord writes its data in RTF format.

InternetExpress allows data sharing at both the selection and document levels. For example, a user may want colleagues to review only certain sections of his document. With only document-level granularity of data transfer, the user must select the relevant sections and copy the data to a buffer, and then paste the data into a new document for transmission. With selection-based transfer, the user need only make a selection and send the data.

Within the Intermedia desktop environment, the data-transfer functionality is integrated into each application, so the actions of selecting and sending the corresponding data work uniformly across all applications.

To achieve this level of integration, the service defines three abstract methods (sendData, receiveData, and understandData) that an application must implement to participate in the service. The service, not each application, is responsible for sending, routing, and storing the data. This paradigm is modeled after the ubiquitous cut/copy/paste paradigm and can be viewed as placing data on and retrieving data from a network-wide, remote clipboard or buffer.

Because many people are familiar with express-mailing packages, our data-transfer service uses an express-package metaphor. The metaphor of expressing a package suggests that users can send and receive multimedia data quickly, reliably, and easily from their desktops.

Our service allows users to share data across heterogeneous environments. We believe this characteristic is crucial, as does Kraut.[9] In a CHI 90 panel address, he argued that any groupware tool that holds allegiance to a single hardware or software base is likely to result in failure. As a consequence, InternetExpress is composed of a desktop-independent component and a desktop-dependent component. The desktop-independent component is a background process, or daemon, that continuously checks and receives incoming data from users on any hardware platform. A user can then receive the data from any desktop that supports our service.

Since each desktop may not have access to the same set of applications, the desktop-dependent component is responsible for choosing a compatible application to present the data to the user. For example, a package of a SuperPaint document from one site could be sent to a remote site that only has MacDraw, since both MacDraw and Super-

Desktop requirements

Before describing a set of desktop requirements to support a data-transfer service, we must agree on a minimal desktop definition.

First, it is necessary to distinguish between functionality provided by an operating system and that provided by a desktop. The desktop provides an extra layer between the user and the operating system. The user perceives this desktop layer in the form of a graphical user interface for accessing and modifying system resources (for example, files, printers).

As the user interacts with the desktop resources, the desktop can monitor and track the activity to provide support above and beyond that of the operating system. For example, the desktop could monitor a user's window layout and restore it the next time the user logs into the desktop.

In terms of functionality, a minimal desktop should have a graphical file manager that allows the user to browse, access, and manage files. More sophisticated desktops will be able to associate applications to data files so that when the user opens a document the proper application is launched to receive and display the document. The desktop should have a set of system commands that are available to the user at all times, usually in the form of menus. Some common commands are printing, opening and closing files, and bringing up system tools.

Environments such as Apple's Macintosh, Sun's OpenWindows, and Microsoft's Windows are examples of desktops that satisfy the minimal definition. A vanilla Unix or DOS environment does not by itself constitute a desktop because it lacks the extra layer of functionality between the user and the operating system resources.

In addition to the minimal definition to support InternetExpress, the environment must be enhanced to handle user identification and user notification. User identification is important, since it is often more convenient to address and receive information on a user basis rather than on a machine basis. The degree of user notification should be commensurate with the importance of the information. For example, extremely important information could warrant interrupting the user by displaying a dialogue box, while less important information could be presented to the user by a visual cue, such as a modified icon. In any case, a user should be able to customize the notification mechanisms used.

In summary, a desktop that can implement our service should have user identification and notification, in addition to the minimal desktop definition described above.

Paint support the PICT data type.

To facilitate sharing multimedia data among users across heterogeneous environments, Straub and Wetherbe[10] and Borenstein[11] argue that applications should conform to industry data standards (for example, PICT, RTF, and WK1). We agree and argue further that a desktop service should not restrict new data standards. If a new data standard is formalized, the service should not have to be upgraded.

InternetExpress supports the transfer of data conforming to any data standard because it is responsible for packaging and unpackaging the information accompanying the data (that is, application type, data type) that describes the selection.

In the "Desktop requirements" sidebar, we enumerate additional desktop functionality needed to implement our service.

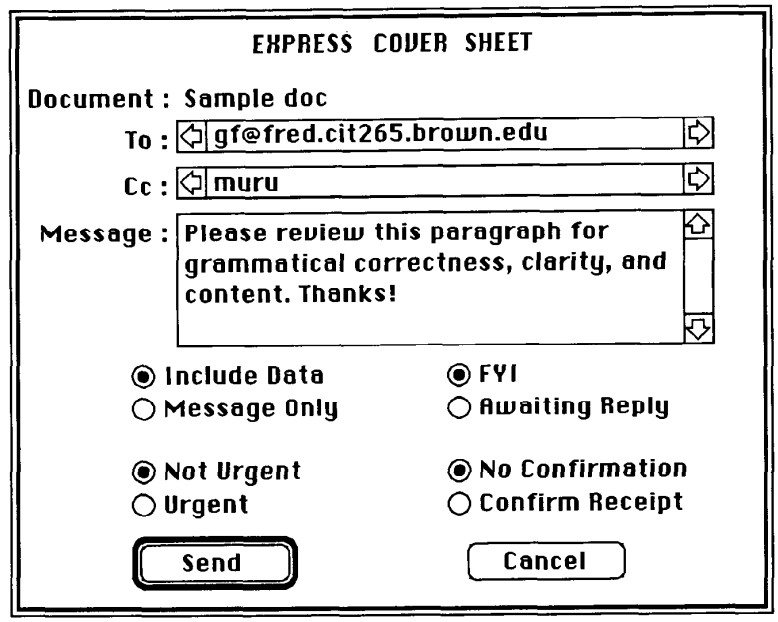

Figure 1. Sample InternetExpress cover sheet.

User interface

Next, we describe how express packages are sent and received using the InternetExpress interface.

Sending an express package. To send an InternetExpress package, a user makes a selection and then fills out an express cover sheet to accompany the data. If a selection is made within an application, the contents of the selection will be sent. Alternately, if a user selects a document icon, the entire contents of the document will be sent. If there is no selection, then only the express cover sheet is sent.

Once a selection is made, the user picks the Send Express Package desktop menu command. This action displays an express cover sheet, as shown in Figure 1.

Various options let the sender quickly classify the status of an express package. The first setting allows the sender to include or exclude the selected data or document. To request the recipient's immediate attention, the sender needs to mark the express package as urgent. The third option is a quick way for the sender to indicate whether or not a reply is expected. To request confirmation when a recipient has seen the package, the sender selects the Confirm Receipt setting. This setting automatically generates and transmits a confirmation notice when a recipient opens, reads, or deletes the package.

Receiving an express package. We have designed — but not yet implemented — an iconic interface for receiving express packages. The icon reflects the state of the spooled express packages, as shown in Figure 2.

A user can receive an express package at anytime. Packages that arrive while a user is not logged on are spooled. When a user starts an Intermedia session, an audible beep and the icon's visual state announce the presence of new packages.

If an express package arrives during an active session, the user will be notified in a variety of ways, depending on the state of the spooled packages. If regular or confirmation packages arrive, the current package icon flashes and changes to reflect the state of the new arrival (see Figures 2c and 2d). If audio notification is turned on, a double beep indicates the arrival of a regular package, while a single beep indicates the arrival of a confirmation package.

For a regular package marked urgent, the package icon (see Figure 2d) flashes periodically in case the user missed the beep. This periodic notification stops when the user responds by viewing the package or by clicking once on the icon. The user has the option of turning off both the beep and periodic notification.

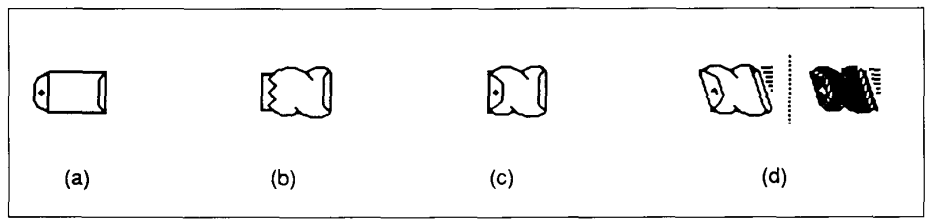

Figure 2. The package icon appears on the user's desktop and indicates the state of the spooled express packages: (a) no packages for the user, (b) one or more read packages are still spooled, (c) one or more new packages exist, (d) at least one new, urgent package exists.

To receive a package, a user selects the Read Express Package menu option or double-clicks on the package icon, which brings up an Express Arrivals bulletin board that displays a list of spooled packages, as shown in Figure 3. The list entries are sorted by time, with the most recently received package at the top.

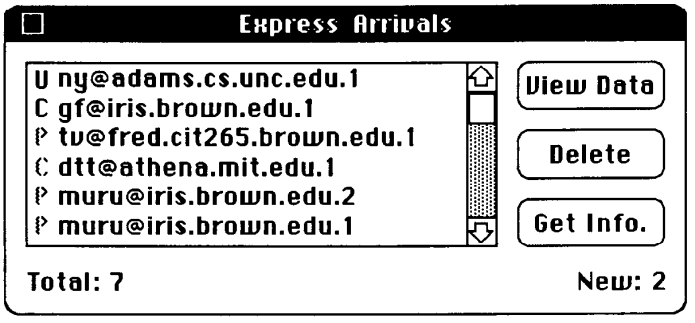

Figure 3. The InternetExpress arrival bulletin board.

Each entry consists of a one-character status indicator, followed by the sender's name. A trailing number helps distinguish multiple packages from the same sender. The indicators are "U" for an urgent package, "P" for a regular package, and "C" for a confirmation notice package. If a user has read a package, the indicator appears in gray.

To operate on a regular express package, a user selects an item from the list and then chooses one of the three buttons: View Data, Delete, or Get Information. The View Data button will retrieve the package data and display it in an application that understands the data type. To delete a package, the user selects the desired item in the list and chooses the Delete button. By choosing the Get Information button, the user can examine the express package cover sheet, as shown in Figure 4.

As a short-cut, the user can double-click on a package entry in the bulletin board list. With this action, the express cover sheet appears, and the package data appears as a new document. Express package documents function like other documents on the desktop. For example, a user can rename and save the package documents into any folder. The cover sheets are discarded when the documents are saved.

Confirmation notices, which look similar to cover sheets, can be opened by selecting a confirmation notice from the bulletin board and choosing the View Data or Get Information buttons or by double-clicking on the entry. This notice indicates the recipient's name, the date the package was sent and read, and the name of the source document.

Architecture

To support the interface described above, InternetExpress uses a TCP/IP (transport connection protocol/Internet protocol) point-to-point connection to send express packages to sites on the Internet. The service consists of a desktop-independent component and a desktop-dependent component.

The desktop-independent component is an express daemon that services a given local area network. The daemon runs at an advertised port on a server machine (analogous to an electronic mail daemon) and is responsible for receiving packages on a user's behalf. Since our service uses the same Internet routing tables as electronic mail, the daemon must run on the same machine as the mail daemon. In this way, InternetExpress is as adaptable to Internet changes as electronic mail.

The desktop-dependent component, the client, defines a protocol for participating applications. It is also responsible for packaging and unpackaging data and for mapping the data to an application. We have implemented the Intermedia desktop-dependent component on the Macintosh family of computers running A/UX, and the daemon on a Sun workstation running SunOS 4.03.

Application programmer's interface. Our Intermedia design philosophy has been to provide general services at the desktop level and to define abstract methods that applications must implement to participate. For instance, linking services in Intermedia define generic methods for applications to participate in the hypermedia functionality.[8,12]

Following our design philosophy, we defined three methods for Intermedia application participation in our desktop data-transfer service:

- DoWriteData,
- DoReadData, and
- DoUnderstandData.

Once a user is ready to transmit, the client invokes the application's DoWriteData method, which returns a data type (for example, PICT, RTF) and data replicating the selection.

When a user is ready to view a package, the client chooses an application and calls its DoReadData method to display the selection.

During application installation into the desktop, the client invokes the application's DoUnderstandData method, which returns all the data types that it can understand.

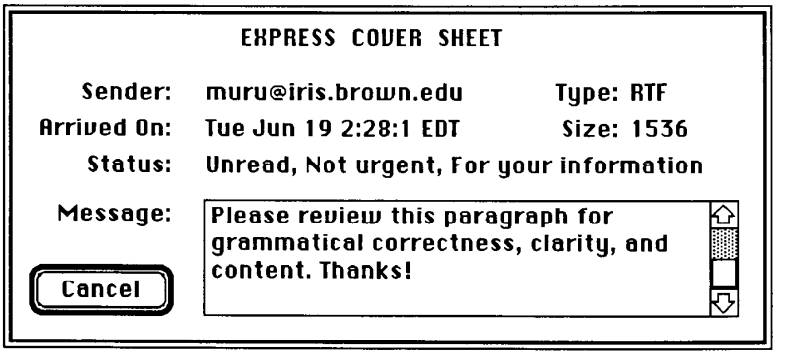

Figure 4. Another sample InternetExpress cover sheet.

Other desktops that wish to implement the express service will need to implement similar methods.

Express daemon. Express packages are not sent directly to a user but rather to an express daemon, which then spools them in the local file system. An express daemon is initialized with a spool directory on the file system that will store incoming packages (equivalent to /usr/spool/mail). Every user has a subdirectory in the spool directory for incoming packages. Each package is saved as a separate file.

An express package file consists of two sections: a header and the application data. The header section contains an application-type field, a data-type field, and a collection of fields representing the cover-sheet information, including a status field (that is, read/unread, awaiting reply/FYI, etc.). The client determines the application type from which a selection has been sent.

The data-type field is set based on the information returned from calling an application's DoWriteData method. The application data corresponds to the actual data needed to reconstruct a selection. As all packages have standard header information, any desktop can examine the package and determine the application for presenting data to the user. The desktop can inspect the package's status field to ascertain its state.

An express daemon accepts connections from any host and provides local area network services that include Internet domain name resolution and package delivery to local clients. To understand the domain name resolution service, consider two local area networks connected by the Internet, as shown in Figure 5.

If Joe wants to send an express package to Ann, his sending process sends it to the local-domain express daemon, which then spools the data for Ann in the file system. When Joe wants to send an express package to Tom, the sending process first obtains the Internet address of the foreign-domain daemon from the local daemon and sends the package directly; the package is then spooled for Tom on his local file system. It is important that the daemon provide this service, since it obviates the necessity for each host to maintain an up-to-date listing of all daemons on the Internet.

As information arrives, the desktop-dependent component or client must be notified. This is achieved by having the express daemon server send a message to the clients that match the recipient's name on the express package. When a user logs into a desktop, the client sends a message informing the express daemon of the user's location. The client-daemon model eliminates the need for client machines to mount a shared express directory and poll for the arrival of new packages. This scheme allows sending packages to clients on heterogeneous machines, since the data is sent using a common communication protocol, TCP/IP.

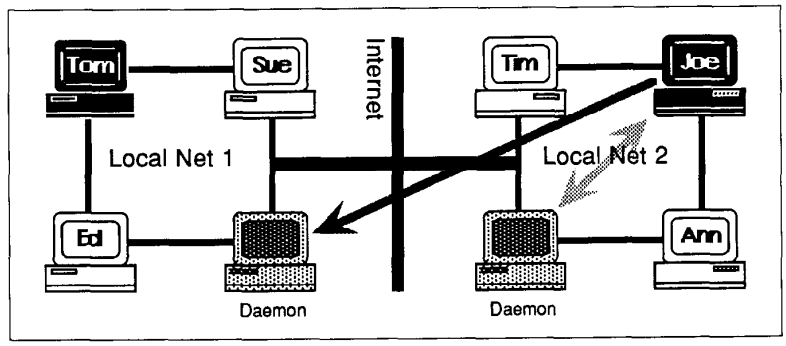

Figure 5. Joe wishes to send an express package to Tom in a foreign domain. The sending process gets the foreign daemon's Internet address from the local daemon and sends the package.

Desktop client database. Each client maintains a database of all the applications that can run on the desktop, together with the data types they understand. Clients who share a file system maintain a shared database. The database information is maintained in two relational tables. One lists data types with applications that understand each data type, as shown in Table 2.

The second table lists applications and their locations in the file system, as shown in Table 3. Several desktops, such as the Macintosh and OpenWindows, have a "finder" mechanism to locate an application in the file system. If such a mechanism exists, the desktop does not need to maintain a second table.

A user usually installs an application into the desktop by dragging an application icon and dropping it into a folder. If the user has changed the location of an existing application, the corresponding entry in the second table is modified. On the other hand, if a new application

Table 2. Data type and applications mapping table.

Data Type	Applications
PICT	MacDrawII, SuperPaint 2.0a
WK1	MS Excel, Lotus 1-2-3
SYLK	MS Excel, Wing Z, Lotus 1-2-3

Table 3. Applications and location mapping table.

Applications	Location in File System
MacDrawII	Applications/DrawProgs
SuperPaint 2.0a	User/muru
MS Excel	Applications
Wing Z	Applications

has been installed, the client adds an entry in the second table and tags it as an application whose data types need to be retrieved and added into the first table.

When a user runs a tagged application for the first time or explicitly selects the Install Application menu command, the desktop launches the application and invokes its DoUnderstandData method. The application is responsible for returning a set of data types that it can interpret. For example, a MacDraw II application returns Drawing, PICT, Color PICT2, and Stationery as its data types. The desktop updates the corresponding entries in the first table.

Mapping data types to applications. When a user is ready to view a package, the desktop client is responsible for receiving the package from the daemon and launching an application that can interpret the data. The client first examines the application type and checks its database to see if the application is available. If the application is available, it launches it; otherwise, the client examines the package's data type and launches the first compatible application that appears in the database.

If no compatible application exists, two possible options are to allow the user to see an ASCII text version or to install a compatible application. We chose not to implement the first option, since nontextual media cannot be readily perceived in ASCII form. Instead, we allow the user to view the cover sheet to recognize the data type and have the option of installing an application that can display the package.

Once an application has been found, the client invokes the application's DoReadData method. This method receives the data and reconstructs the selection.

User evaluation

Although our prototype has had limited use, we have gathered some initial feedback on the system. Users reacted positively regarding the system's ease of use and simple interface. Defining a selection within an application and being able to send it to a recipient was commonly thought of as an extension to the cut/copy/paste metaphor (that is, cut/copy/paste/send).

In terms of functional extensions to the system, a few users wanted to be able to send folders or even entire disks. This is a bit tricky, since there is not really a folder-viewing application to invoke. However, the desktop could take on some of the responsibility to display the folder.

Viewing a package would require the desktop to construct a temporary folder. If a user decided to save the new folder, the desktop would ask the user where to place it relative to the existing folder hierarchy. The InternetExpress service could easily extend the header definition of packages to support folder packages.

A recursive package header definition would allow folders within folders

to be sent as one package. Similarly, if a user wanted to send a set of documents having different data types, a virtual folder would be constructed and sent to the recipient.

Some users wanted to be able to send a user face or company logo (in the form of a bitmap) as part of the cover sheet information. We could easily do this in InternetExpress by sending the bitmap data as part of the header information.

Although Intermedia has a small set of applications, users recognized that they would want to specify preferred viewing applications. That is, if more than one application can handle a particular data type, the resolution algorithm should check the user's set of preferences for selecting the viewing application.

In InternetExpress, we have designed a desktop data-transfer service and prototyped it within the Intermedia desktop environment. Based on our design and implementation, we recommend the following characteristics for a similar service in other desktop environments:

- Provide a simple application programmer's interface to allow desktop applications to easily participate in the service.
- Allow applications to register the data types they understand with the desktop so that the transfer service can automatically choose an appropriate application for presenting the incoming data to the user.
- Provide a daemon to receive incoming information on behalf of users so that they can view the data at their convenience from any workstation on the network.
- Support transfer of not only selections within a document but also any desktop selection.
- Have a wide range of notification mechanisms available on the desktop to inform the user of various events.

We believe that InternetExpress solves a real user need, provides a simple user interface, and transfers data quickly and at a low cost to the user. We also suspect that, with widespread adoption of the InternetExpress service, users would share multimedia data as frequently and conveniently as they do electronic mail. ■

Acknowledgments

We especially thank Nicole Yankelovich for her help in refining the concepts described in this article and for her valuable comments on early drafts. We also thank Norman Meyrowitz and Karen Catlin for their valuable critique of the manuscript, and we acknowledge Helen DeAndrade for her assistance in preparing the diagrams.

References

1. T. Crowley, "How Can We Make Groupware Practical?" panel, *Proc. 90 Conf. Computer-Human Interaction*, ACM, New York, pp. 87-89.
2. G. Salton, *Automatic Text Processing*, Addison-Wesley, Redding, Mass., 1989, pp. 471-473.
3. J. Johnson et al., "The Xerox Star: A Retrospective," *Computer*, Vol. 22, No. 9, Sept. 1989, pp. 11-29.
4. B. Shneiderman, "Direct Manipulation: A Step Beyond Programming Languages," *Computer*, Vol. 16, No. 8, Aug. 1983, pp. 57-68.
5. N. Yankelovich et al., "Intermedia: The Concept and the Construction of a Seamless Information Environment," *Computer*, Vol. 21, No. 1, Jan. 1988, pp. 81-96.
6. M. Satyanarayanan et al., "The ITC Distributed File System: Principles and Design," *Proc. 10th Symp. Operating System Principles*, ACM, New York, 1985, pp. 35-50.
7. K. McConnell, D. Bodson, and R. Schaphorst, *Fax: Digital Facsimile Technology and Applications*, Artech House Books, Norwood, Mass., 1989.
8. D. Crabb, "Mac Goes Fax," *Byte*, Vol. 14, No. 5, May 1989, pp. 208C-208H.
9. B. Kraut, "How Can We Make Groupware Practical?," panel, *Proc. 90 Conf. Computer-Human Interaction*, ACM, New York, pp. 87-89.
10. D.W. Straub and J.C. Wetherbe, "Information Technologies for the 1990s: An Organizational Impact Perspective," *Comm. ACM*, Vol. 32, No. 11, Nov. 1989, pp. 1,328-1,339.
11. N.S. Borenstein, "Multimedia Electronic Mail: Will the Dream Become a Reality?" *Comm. ACM*, Vol. 34, No. 4, Apr. 1991, pp. 117-119.
12. M. Palaniappan, N. Yankelovich, and M. Sawtelle, "Linking Active Anchors: A Stage in the Evolution of Hypermedia," *Hypermedia*, Vol. 2, No. 1, Summer 1990, pp. 47-66.

Murugappan Palaniappan is a senior software engineer with Aldus Corp. in Seattle, Washington. Previously, he was a research software engineer at Brown University's Institute for Research in Information and Scholarship and the principal architect of the institute's intelligent agent research program. He played major roles in the design and development of the Intermedia system. His major research interests are in the areas of intelligent agents, component-based architecture multimedia systems, and computer-supported cooperative work.

Palaniappan holds an MS degree in computer science from the University of North Carolina at Chapel Hill, an MS degree from Duke University, and a BTech degree in civil engineering from IIT Madras. He is a member of the ACM, SIGOIS, and the IEEE Computer Society.

George Fitzmaurice is a PhD candidate in computer science, working in the Dynamic Graphics Project at the University of Toronto. Previously, he was a research software engineer at Brown University's Institute for Research in Information and Scholarship and a senior researcher and architect of the agent project. He and made substantial contributions to the Intermedia system. His major research interests are in the areas of computer-supported cooperative work, human-computer interaction, and multimedia communication.

Fitzmaurice received his ScM in computer science at Brown University and his ScB in mathematics and computer science at MIT. He is a member of the ACM, SIGCHI, and the IEEE Computer Society.

Readers can contact Palaniappan at Aldus Corp., Engineering Dept., 411 First Ave. South, Seattle, WA 98104, Internet muru@aldus.celestial.com; and Fitzmaurice at the University of Toronto, Dept. of Computer Science, Toronto, Ont., Canada M5S 1A4, Internet gf@dgp.toronto.edu.