A large-scale computer conferencing system

by D. M. Chess
M. F. Cowlishaw

This paper discusses the relationships between computer-mediated communications and other forms of communication and describes a particular computer conferencing system in use within IBM. The system described is quite large, with over three thousand contributors and over twenty thousand readers. We discuss the structure of the system, the actions that users can take, and the ways in which the system is being used. Neither the definitions presented nor the system described are intended to be the last, or only, word on the subject; as computer-mediated communications and distribution become more and more important in the business and professional communities, we will need more ways of thinking about communication systems and about information distribution in general.

When computers were thought of primarily as devices for manipulating numbers, they were used mostly by scientists and a few others (such as census takers). They were also used for applications that mostly lent themselves to FORTRAN and other formula-oriented programming languages.

A slightly more sophisticated use of the computer takes advantage of more of the inherent flexibility of the machine in record-oriented applications, such as payroll management, that are generally written in COBOL and similar languages.

In recent years, the growth of modern languages (such as REXX), which enable computers to store and process *general information* rather than simply numbers, has led to a broad-based revolution in computer usage, and brought computers into areas that have little use for numerical calculations, but have a critical need to automate the generation, storage, and *communication* of information. There is no reason to think that this revolution will not continue.

Even simple electronic mail, the most primitive form of computer-mediated communication, can have a huge impact on (for instance) a business or a scientific community. Written notes, instead of taking several days to cross a country (or a county), may be received within minutes of being sent. This short time gives written communication almost the immediacy of a telephone call, while retaining the advantages of the written medium (written messages may be more carefully thought out, are easily filed, and do not require the recipient to be available when the sender composes the message).

The use of electronic mail is one very small part of computer-mediated communications. Given the vast capabilities of the computer for general-purpose information processing, the use of computers as communications intermediaries has the potential to start a communications revolution fully as significant for the future of business and industry as was the first industrial revolution.

This paper describes one particular exploration of the possibilities of computer-mediated communications—computer conferencing. It describes an on-

going project, which takes advantage of only a small part of the communication potential of the computer. Even so, it has had a major impact on the way thousands of people get their jobs done. We hope that this description will be useful to those already in the field of computer communications, and also to those many whose lives and work will soon be touched by it. We also hope to provide enough information to help those who are actually designing or building conferencing systems of their own, in any electronic environment.

Conferencing and communication

Since computer conferencing is a type of communication, we first briefly discuss how it compares to other sorts of communication and what new features it has. We will use the term "computer conferencing" to refer to all human communication that involves a computer. Systems that do computer conferencing are often called "computer-mediated communications systems," or "CMCSs."

Consider the advantages and frustrations of communication by telephone. On the positive side, a telephone conversation (when the other party is in, and the other phone is not busy) can be set up in less than a minute, and business can be conducted at the full speed of dialogue. On the negative side, it is impossible to communicate at all by telephone if the person you are trying to reach is not available at the time you want to communicate. Anyone who depends on the telephone to do business knows how often "telephone tag" occurs, in which each party leaves message after message for the other, sometimes going weeks before any useful communication actually occurs. Telephone conversations are also somewhat ephemeral; unless special arrangements are made to tape the discussion, any information exchanged is available only in the memories of the parties involved.

Mail is a very different form of communication. The communication rate (one letter in each direction every few days) is much, much slower than a telephone conversation. Nevertheless, you can write a letter to someone with the knowledge that the information in it will be received, even if the person it is sent to is not available as you write it or when it is delivered. And (unless it is destroyed) the letter will still be available for rereading or for copying to others long after it is written. We say that the telephone conversation is "synchronous" because both parties have to be available at the same time, whereas the letter is "asynchronous" because they do not. The telephone conversation is relatively fast, whereas the letter is slow. The letter can be "retained," and the telephone conversation typically cannot be retained.

Electronic mail, the simplest form of computer conferencing, offers the advantages of both of these kinds of communication; an electronic message can cross a continent in minutes, but if the recipient is not available when it arrives, it will wait. Thus, little time is lost through delivery delays, and yet there is no danger of "telephone tag." If either the sender or the receiver wishes, an electronic message can be saved as data for later printing, for sending on to others, or for use in composing documents or other notes. In the terms we are defining, electronic mail can be a fast, asynchronous, retained-information medium.

The telephone, the mail, and electronic mail are all best suited to contacting a specific individual, when you know exactly who it is you are trying to reach. It is often necessary, or desirable, to reach people you do not know by name or address, people who, for instance, have specific skills or knowledge that you need. This function is performed by (among other things) newspaper want-ad columns. A want-ad is a message to anyone who may read it, saying something like "if anyone has this particular thing, contact me." Computer conferencing can offer an improved version of the want-ad; in a suitably designed computer conference, it is possible to ask questions of a large group of unknown people and receive an answer within hours or even minutes. It is also possible to change the content of a question if it was unclear in the first place and to do other things that are difficult with newspaper want-ads. Full computer conferencing, then, as well as offering fast asynchronous retained communication, can also offer access to a large pool of people.

In addition to comparing computer conferencing to more traditional media, it is also good to have ways to compare different computer conferences to one another. One way to do this is by looking at the kind of information that can be transmitted. We usually think of communication as taking place through language, by the transmission of words from one person to another. Another useful ability (which might be called "information distribution" rather than communication) is to transfer other types of information, such as images and computer programs. This ability is not communication in the usual sense, since it is not an exchange of words, but it is something that many computer conferencing systems can do, and it can be very desirable.

One last distinction between conferencing systems involves the environment that the user is in when using the system. As computer usage becomes more commonplace, users of computers will tend to develop familiar environments in which they do most or all of their computer-related work. Computer-mediated communications systems may be characterized by whether or not they allow the communicators to remain in their home environment. Systems that require a direct connection to a particular computer, for instance, contrast with systems that may be accessed from a number of computers and working environments. We will call the former "closed-environment" systems and the latter "open-environment" systems. This distinction is not always well-defined, of course. One communicator's familar working environment may be foreign ground to another.

As computer usage becomes more commonplace, users will tend to develop familiar environments for their work.

Computer-mediated communications systems of all these types exist today. The many private "bulletin-board" systems are typically asynchronous, retained-information systems that allow distribution of any sort of digitized information (generally words and programs) among any number of persons. These systems often have synchronous components (like the "CB" systems of CompuServ™), through which communicators who are accessing the system simultaneously may "chat." These systems are primarily closed environments; they expect the user to be working at a terminal directly attached to the host computer. If other computers are used in the interaction, their role is generally minor: serving as terminals or passive recorders of information.

In talking about specific computer conferencing systems, it will be useful to describe explicitly the *entities*, *attributes*, and *actions* that make up the system. By making these categories explicit, we find it easier to compare different systems and to determine the actual capabilities of a given system.

To be usefully called an entity, a component of the system should have attributes and be affected by actions. A message is an entity in a typical bulletin-board system, but a single line of a message is not, since no commands operate on a single line. If there were a command "display line n of message m," we would be more likely to speak of lines as individual entities. The entities that the system recognizes will affect the way in which the users use and perceive the information presented. If, for instance, lines of a message are accessible as entities, users will be likely to follow this structural lead in composing their messages, and conventions (such as "put the topic on line 1") will tend to arise. User expectations about extensions to the system will also reflect the current structure. If lines are not system entities, for instance, users may be less likely to ask for a command to copy a given line of a given message into a new message that is being composed.

The rest of this paper describes a fast asynchronous retained-information system that may be used to distribute any digitally encoded information among any number of persons. It is to a large extent an "open-environment" system in that it runs on a network of computers and most of its users communicate with the system across the network from their normal working environments.

TOOLS information distribution

History and current usage. The conferencing system we describe in this paper is called TOOLS. It was originally conceived and written by Cowlishaw in 1981 as a relatively simple file server system to allow secure and controlled shared access to collections of software tools. When the IBM Personal Computer (IBM PC) was announced, internal interest in the product was high enough that a means of widespread discussion on the topic of the IBM PC was highly desirable. A modification of TOOLS with some limited conferencing features was created by researcher Walt Daniels to perform this function. This "IBMPC" conference grew very rapidly and proved enormously popular; it is still the largest TOOLS-based conference in IBM. Interest in conferencing spread, and more CMCS features were added to TOOLS. There are currently 790 TOOLS systems within IBM, at least 100 of which are used primarily for computer conferencing. One developing pattern is to have a TOOLS system

used mainly for conferencing (for instance, IBMPC and IBMVM, a conference devoted to the Virtual Machine/System Product, or VM, operating system) and another system devoted to the distribution of related software. (There is a PCTOOLS system corresponding to IBMPC, and a VMTOOLS system corresponding to IBMVM.) TOOLS-based conferencing has added enormously to the availability of information in all these areas and boosted the productivity of thousands of IBM employees.

The basic entity in a TOOLS system is the file.

This section describes the entities in the TOOLS system, their attributes, and the actions that users can perform on them. The implementation of the system and our experiences with it are discussed later.

Basic TOOLS entities and attributes. The basic entity in a TOOLS system is the *file*.[1] Each file may contain a single item (such as documentation for a program or a report on some topic) or a series of items, potentially from a number of different contributors.

Each file in a TOOLS system has (in addition to the unstructured information in the file itself) a name and a type (each from one to eight characters in length), an owner (some user of the system), a description (a short summary of what the file contains), a "last change time" (which records the time and date of the last alteration made to the file), a length (in lines), a size (in kilobytes), and a few other attributes that we mention later.

As TOOLS was originally designed, a file could be changed only by its owner (or by a specially privileged user). With this restriction, true conversations could not occur, since conversation requires at least two speakers. When TOOLS began to be used for conferencing, features were added to allow more than one person to update the same file.

It is now possible to specify to TOOLS that files of certain types (as reflected by the file type attribute) will consist of a series of items and that any user satisfying certain criteria should be allowed to add a new item to the file (an action usually called "appending"). These files therefore consist of a series of items making up a discussion among a group of people on some topic. Such files are often known as "forum" files (or "forums" or "fora"), and the file type "FORUM" is often used for this purpose. Each item in such a file has attributes giving the time and date it was created, the file to which it belongs, and the identification of the user who added it. When TOOLS is used primarily for computer conferencing, most of the files on the system are of this sort.

When a system accumulates a very large number of files, it can be difficult to keep their relationships in mind. This problem is especially difficult when the underlying operating system restricts the possible set of file names. For instance, a single program might consist of several program files and several documentation and reference files, and if the operating system allows (for instance) only 16 characters in a file name, it can be hard to find a set of names that makes explicit the fact that all the files belong together. To address this problem, the concept of a *package* was added to TOOLS.

A package is a collection of files, one of which (the "package file") serves as a set of pointers to the others, and contains in addition information about the package itself. Packages are useful, for instance, for keeping together the elements of a system of programs, grouping related documents on a single topic, or showing explicitly that one group of files is related to another as a prerequisite. For example, on IBMPC all the papers from the latest internal symposium on the IBM Personal Computer are grouped as a package and may all be requested with a single command. Users may perform some actions on an entire package without necessarily knowing what files it contains. When a user requests ownership of a package file, ownership of the files listed in that package is automatically requested as well. Packages may contain pointers to other packages, and a user requesting a package will also be sent any packages that it points to. For each file on the system, TOOLS keeps track of what packages (if any) it belongs to.

Packages provide a means for grouping relatively small numbers of files that belong together and that should be treated as a unit for many purposes; this imposes a fine-granularity organization on the files

in the system. It is also desirable to impose broader sorts of organization. The *disk* is the large-scale organizing concept in TOOLS; a single disk might contain all the discussions and documentation about microcomputers, whereas another might contain all the information about mainframes.

In more detail, a disk is a group of files and packages, along with history and usage information, logs for audit purposes, and other associated information. The term *disk* comes from the implementation; a reasonable nonelectronic analogy for the "disk" is a single drawer of a filing cabinet. (We will have something to say later about the dangers of this type of analogy.) A single TOOLS system may manage one or many separate disks. Shadowing (which we will describe later) allows specified disks on one TOOLS system to be automatically copied to a number of other systems to make access more convenient for distant users. Most TOOLS commands apply to a specific disk on the system being addressed.

TOOLS itself does not interpret the data in its files.

TOOLS does not interpret the information contained in a file, except for those entries that consist of a series of items. This has important consequences for the usefulness of TOOLS for nonconferencing information distribution. Since there are no requirements that "subject lines" or other attribute information actually reside in the file, TOOLS may be used to retain and distribute programs, digitized images or audio data, or anything else that the host computer can store and transmit.

The fact that TOOLS itself does not interpret the data in its files also means that there is no way to, for instance, hold a secret-ballot election using only built-in TOOLS actions. TOOLS does have, however, a number of "user-exit" points, so that a more structured interpretation of incoming data may be added without modifying the system itself. The addition of structure can be helpful in tailoring the system for specific tasks and for reducing information overload. We will have more to say on these topics later.[2]

The users of a TOOLS system have attributes describing the actions that they are permitted to perform, the files that they own, and the files and packages that they are interested in. As we will describe in more detail, there is a TOOLS command that allows users to "subscribe" to a file and automatically receive any updates that are made to it. For files that are particularly interesting, this command saves the user the trouble of manually checking for updates.

As in any system that will be used for a variety of purposes, there is often a need to control the actions that users are authorized to perform. Since TOOLS is intended to be able to handle very large groups of users, it is possible to control authorization levels for many users at a time; that is, the person setting up a TOOLS system may specify (for instance) that all users shall have a certain privilege level, except that users at a certain location shall have a different one, and some specific users shall have still another. The granularity of control is very fine; for any user, it is possible to specify precisely the set of actions that that user should be able to take with regard to each file type on the disk. In contrast, commonly associated sets of actions are grouped into convenient synonyms, so if the system is being used in a typical way, the authorization list will be small and simple to maintain. Some uses of the authorization mechanism are described later, and details are given in Appendix A.

Shadows. In a large distributed network, the time it takes to access data may be strongly affected by where those data are located. For a user on a network that spans continents, data actually resident on the local system may be immediately available, whereas data on a distant node, although accessible, may take some time to arrive. To address this problem, TOOLS supports various mechanisms for maintaining local copies of the data contained in a disk.

These mechanisms are implemented by allowing some users of the TOOLS system to in fact be other TOOLS systems. For instance, the system maintains a list of users to whom are forwarded copies of every request that causes a change to the files on a particular disk. This list is commonly used to maintain "shadows" of particular disks at remote TOOLS locations; subject to communications delays, the systems to which the requests are copied can maintain duplicates of the data in the master TOOLS system.

The status of these shadows may vary. A shadow may be a *passive slave*, in which case it blindly accepts updates from the master but cannot itself generate updates; it may be a *servant*, a shadow that accepts updates from its master but also accepts from users update requests that will be copied back to the master (and hence to all the other shadows); or it might be a *peer*, a TOOLS system that is the equal of all its peers—any peer system may accept requests from users and copy that request to all its peers. A single TOOLS system may use these linkages in any combination. The same system can be the peer of several other systems, pass on copies to passive slaves, and accept updates from a master that is in some sense at a higher level in the hierarchy of systems.

TOOLS does not offer some features that are often found in closed-environment conferencing systems. In particular, there is nothing corresponding to a "notebook" (a collection of data that may be accessed and modified only by the user whose notebook it is) or a "personal message" (a piece of data sent from one user to another and not accessible to anyone else). Because TOOLS is an open-environment system, these facilities are already available to the users in their home computing environments (notebooks as files in the local operating system, and personal messages as electronic mail via the network that TOOLS runs on) and need not be resupplied by the conference.

Basic TOOLS actions. Here we discuss the most important actions that users of the TOOLS system can perform. A detailed list of the general user actions is given in Appendix B.

The first command that a new user might issue to a TOOLS system is "help." At the option of the system maintainer, each disk that a TOOLS system maintains may have some "help" information associated with it. The help command returns this information for each disk that has it, as well as general help information for the system as a whole. Disks that should not be generally visible to the public may be hidden from the help command by simply not including any help information for them.

After an interesting disk has been found, either through the help command or from some other source, the next command might be "summary." For each file in the specified set of files (including "* *" for all files), the summary command returns to the user the name and type of the file, the identification of the owner of the file, and a short description, originally entered when the file was created. Using this information, the user can determine which files are most likely to be interesting or to contain the information sought.

To get one of the desirable files just identified, the user might next use the "get" command, which simply ships the user a copy of the file or files named. Options on the command can be used to request only those items in a file beyond a given date or only those files in a set that have been updated since a given date. Local users of the system (described later) may also be able to access the data in the system directly, without using any TOOLS commands.

Having identified and read one or more files on the system, the user may wish to keep up with the file by being informed automatically of any updates to it. The TOOLS "subscribe" command may be used to do this. The "inform" command, related to "subscribe," may be used to request a note saying that the file has been changed, rather than a copy of the change itself, or to request notification of any new files that are created.

The "create" command is used to create a new file on a TOOLS disk. The user supplies the file itself and a short description (used in replying to "summary" and similar requests). A user who creates a file is automatically the owner of that file; ownership may later be transferred to another user with the "newown" command. A user creating a *package* file is also given ownership of all the files named in the package (subject, of course, to someone else already owning a file with that name, and to the user's authorization level). Other commands to manipulate and query file ownership are described in Appendix B.

Having created a file, the user may make a new version available with the "replace" command or delete the file and give up ownership with "erase."

The owner of a file, and anyone else so authorized, may add an item to a file with the "append" command. This command is also used to modify items already contributed. The "append" command is the most common method of contributing to TOOLS-based conferencing systems; the IBMPC conference processes more than 300 append requests in a typical business day.

These actions are only a sample of the most commonly used TOOLS commands; see Appendix B for

more details. Every command issued is verified before being carried out. TOOLS matches the user's identification against the authorization lists for the system and the disk and only carries out the action if it is possible and permitted.

Every file on the disk is contained in some package.

Some uses of the authorization levels. Appendix A describes the authorization mechanism in some detail; here we give a few examples of how it can be used.

The typical "open" conferencing system in IBM is set up to allow any user with access to the system to create a new file, to subscribe to files, and to get any file. In addition, all users are authorized to add items to files with certain file types (typically including "FORUM," "BUGS," and so forth). This simple authorization structure requires only two lines in the TOOLS control file.

A typical software repository system might specify that all users should be allowed to get files, subscribe to files, and create new package files, but not to create new nonpackage files not mentioned in any package file. This specification means that every file on the disk (except the package files themselves) is contained in some package, and it is possible to find out with what package (and therefore what piece of software) every file is associated.

It is also possible to allow all users to read the information in the system but to allow only certain users to add to the information. This could be used to implement an "official news" system or a repository of product documentation.

Some discussions call for a still tighter level of control. If, for instance, it is necessary to restrict access to the system to a specific list of users, TOOLS can be told that, for a particular disk, only specified users should be able to access the disk at all. Each of the authorized users may be given any desired level of access; some users may be allowed only to read the information, others only to add to existing files, and still others to create new files.

In all these cases, one or more users may be given special privileges, enabling them to perform any actions on any files on the disk. This procedure is necessary to allow for maintenance in case of system problems, and for effective enforcement when a discussion threatens to stray outside the bounds of the purpose of the conference. Details of all these privilege levels are given in Appendix A.

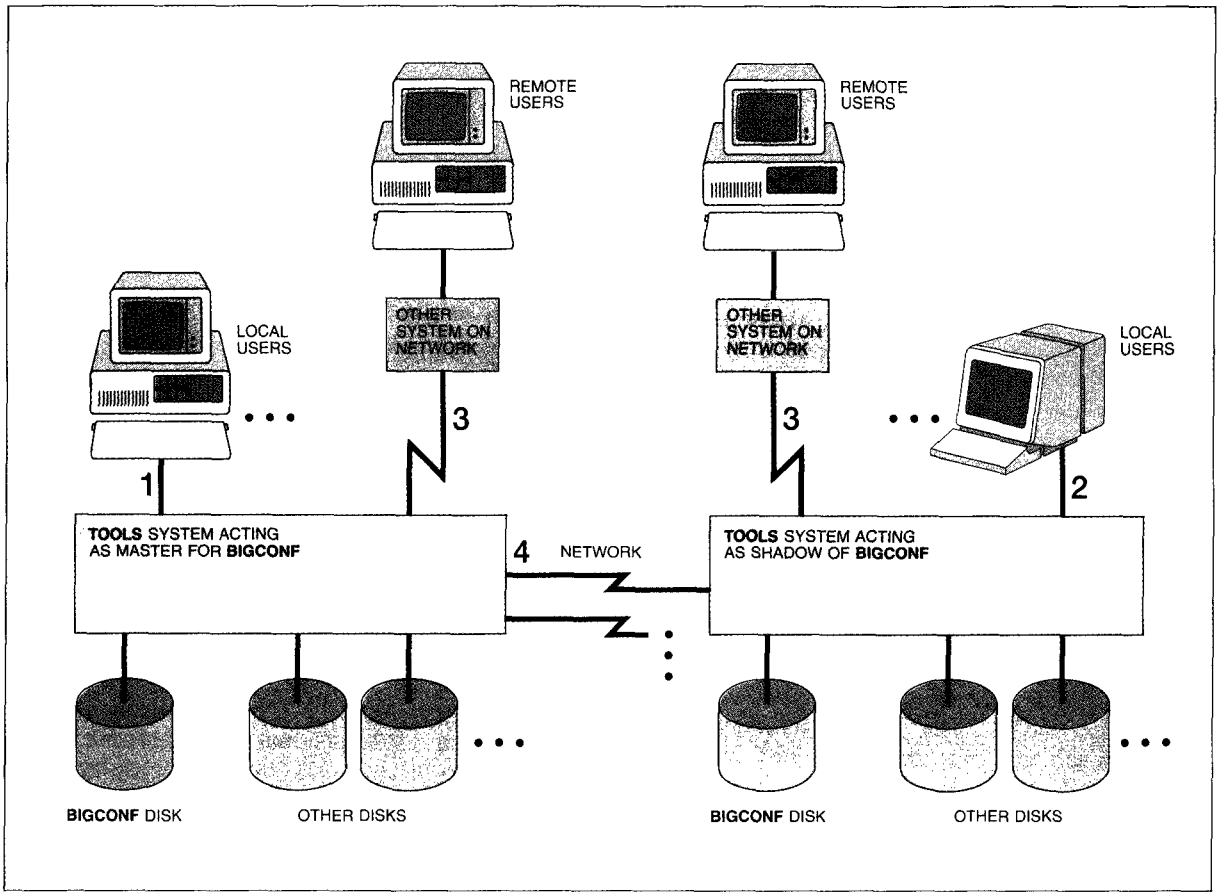
Experience with TOOLS

The current implementation. The basic entities and actions in the TOOLS system could be implemented in any networked electronic environment. Except perhaps for the notions of file name and file type, and some other small details, the structure of the system is independent of the underlying operating system and hardware. This section will describe the current implementation as an example of actual use.

TOOLS is currently implemented in the REXX language,[3] running under the Conversational Monitor System (CMS),[4] in an unmodified Virtual Machine/System Product (VM/SP)[5] virtual machine. Each user in this environment has control of an electronic workplace (called a "virtual machine"), and communications and data manipulation take place through this workplace. For historical reasons, the user's data file storage area is called a "minidisk," the incoming electronic mail box is a "virtual card reader" (or just "reader"), and the outgoing mail slot is a "virtual card punch" (or just "punch"). Files that are in the process of being transmitted from one user's workplace to another's are sometimes referred to as "punch files." Some minidisks are available for access to all users on the same physical computer (or at the same location) via the "LINK" command. A user's primary minidisk is called the "A-disk."[6]

Each TOOLS system runs in a single virtual machine. The virtual machine has one minidisk used for the system itself and various audit and overhead files, and one minidisk for each of the TOOLS "disks" maintained by the system. The "files" in each disk correspond to CMS files; "items" within a file correspond to sequences of lines, delimited by specially formatted and easily recognizable separator lines, which contain the location and user identification of the contributor and the time and date (in GMT) that the item was received.

Figure 1 A sample TOOLS configuration, showing how users may access data on a TOOLS disk called BIGCONF



TOOLS systems in this implementation communicate with their users and with one another through virtual card readers and card punches. Communication between virtual machines resident on different computers is accomplished through the Remote Spooling Communication Services (RSCS)[5] networking system, a store-and-forward system that connects various types of mainframe computers. Efficient use of the RSCS network is greatly enhanced by experimental modifications which allow file fan-out: If a file is sent simultaneously to a large number of destinations, only one copy of the file will traverse any single link in the network. The extremely rapid growth in conferencing that has occurred in recent years would probably not have been possible without this modification.

In many cases, local users (users with virtual machines on the same physical computer as a TOOLS virtual machine) access the data of the system simply by LINKing to the minidisks on which the data reside. Local users may also communicate with the system through the virtual reader and punch or (if the TOOLS system is set up to allow it) through the Inter-User Communication Vehicle (another way that VM users and facilities communicate, similar to the reader and punch in function but different in detail).

A sample TOOLS configuration is shown in Figure 1. Users at the location where the master TOOLS system for BIGCONF (1) or one of its shadows (2) resides may simply LINK to the proper minidisk. Other users may communicate with the master or a shadow through the network (3). The master and the shadows also communicate through the network (4). Any of the TOOLS systems pictured may also maintain other disks as masters, slaves, servants, or peers.

The TOOLS system itself consists of one large (about 8000 lines) REXX program, some small subsidiary

REXX programs, and a small assembler-language program that is optionally used to attain higher speeds for some common requests. Audit trails of requests received, errors or anomalous conditions in the system, and system performance statistics are kept in three files on the A-disk of the TOOLS machine. History files and some other overhead files for the individual disks are kept on the maintained minidisks themselves. The authorization information and other control information are kept in a file on the A-disk and read into virtual memory when the main program is invoked.

In a typical TOOLS system, the TOOLS virtual machine is automatically logged on as part of the system-startup procedure on the host computer. It reads the control and authorization information from its A-disk, links to the disks it will be maintaining to perform some startup chores, and then waits for requests to appear in the virtual card reader. When requests arrive, they are read and logged, the authorization of the user is checked against the action requested, and (if all tests are passed) the action is performed. If the action is a read request, the requested information (generally files or parts of files) is returned to the user as a file which will eventually arrive in the virtual card reader. If the action is a write request, the requested changes are made to the data, users subscribing to the changed files are sent the changes, and the user initiating the request is sent a confirmation message. For many actions that alter the content of the data base being maintained, a backup copy of the data is kept for possible later use of the REGRESS command.

Given a TOOLS-maintained disk that has shadows at several locations, there are four basic ways that a user can access the information in the files on that disk:

1. If there is a single master location at which the disk resides, users at that location can access the information very easily (typically with a single "link" command) and be certain that what they are seeing is the latest available.
2. Users not at the master location may access the latest information by sending GET and LIST requests to the master system or by subscribing to files that are of interest to them.
3. Users at locations with peer, servant, or slave shadows of the disk may access the data of the shadow directly (by linking to the shadow copy).
4. Users at locations where no shadow of the disk exists may send requests to a nearby shadow.

The first two methods guarantee that the data accessed will be current at the time the data are sent. Because of network delays, data from nonmaster shadows will not always be completely up to date, although the difference will not be significant for typical conferencing applications. The first and third methods provide instant, easy access to the information, whereas the second and fourth are more asynchronous. (A read request is issued, and some time later the requested data appear in the user's incoming electronic mail box.)

Users may access a TOOLS virtual machine through any program that produces the proper punch files.

User interfaces. Since TOOLS communicates through virtual punch files, users may access a TOOLS virtual machine through any program they choose, provided that it can successfully produce the proper punch files. A REXX program (called "TOOLS EXEC") is generally used when communicating with a TOOLS system from other systems running CMS. It may either be called directly by the user or used as a "back end" by other programs that present the user with a different interface but call TOOLS EXEC to actually send the request. By centralizing details of the exact syntax of requests, the position and order of arguments in request decks, and other minutiae, the TOOLS EXEC frees interface writers from worries about possible future changes to these details, since any such changes will be hidden (as much as possible) within the TOOLS "back end."

Interfaces to the TOOLS system have also been written (or are in the process of being written) for other operating systems that run on networked machines, including Multiple Virtual Storage (MVS),[7] the System/38,[8] and workstation operating systems running on workstations attached to the main network through local-area networks.

Given the availability of "back-end" programs, users (or user support groups) are free to experiment with

novel user interfaces and to fit the TOOLS user interface into local styles and standards. There are TOOLS interfaces which are entirely command-line oriented, others which operate through menus with large amounts of prompting and "help" information instantly available, and others in which the CMCS appears to the user as a (relatively) integral part of a uniform interface system that also includes electronic mail, document preparation, and other functions. This flexibility is not entirely free, of course. It is not always possible for one user to instruct another user (perhaps from another site) on how to access the system, since the locally implemented interface methods may be different. The common back end guarantees a method of last resort, though, since anyone who knows how to use the TOOLS back end itself can invoke it directly, regardless of locally designed interfaces.

The conferencing environment. To give some idea of the scale of usage that TOOLS is designed to handle, we will present some experience with the IBMPC facility (currently managed by Chess). There are 103 "first-level shadows" of the system (shadows that receive copies of all data from the master system) and at least 50 "indirect shadows" (shadows that receive copies from shadows). The master system handles roughly 2500 requests per business day.

There are upwards of 3500 files on the IBMPC disk, and recently a second disk has been added to hold old and "archival" information. On a typical business day, there are more than 300 updates to the files on IBMPC, the large majority of these being APPEND requests. The total amount of information on IBMPC is about 90 megabytes. It is difficult to estimate the number of people who read the information in the system, but it is more than 20000 (and possibly as high as 90000; this number is hard to determine). The number of contributors is over 3500.

Contributors to the data base are from a relatively wide range of backgrounds. Early in the operation of the system, before any novice-level interfaces had been written, the only participants were people with experience in CMS, who had the ability to deal with sometimes-fractious systems. They were primarily programmers and hardware developers. Most of the information in the system was at a very technical level. As use of the system has become simpler and knowledge of it more widespread, the user mix has become more varied, and contributions now come from the administrative, sales, and other nontechnical areas.

Information and overload. It is impossible to classify completely the mass of data in such a large system. Some of the most common contributions to the discussion, however, include

- Appeals for assistance—A user trying to get some piece of software to work in a slightly odd environment, or to make some piece of hardware behave slightly differently, or just trying to understand some particularly turgid paragraph in a manual will often contribute a question. A similar type of question comes from users who need to accomplish some specific task and are looking for a program or technique to do it. Such questions are generally answered within an hour or two, if the hardware or software concerned is widely used.
- Tips and techniques—Often begun in answer to questions of the first sort, "how to do it" discussions are frequent in the system.
- Bug reports and enhancement suggestions—Users of any software on any computer system will find things that the system does wrong, or things that they would rather have it do differently. One common contribution to the data base is the combined bug report and "wish list."
- Product information and reviews—General information about software or hardware that much of the community may not have seen is also included. IBM announcement letters for products related to the Personal Computer are placed in the data base, and hands-on reports are often put out as well.

Information overload is of major concern in such a system, and various means of combating it are under study. An *ad hoc* group of interested persons has begun to edit and concentrate the sometimes rambling and chatty FORUMs into more structured technical notes, which are written by several people, reviewed for accuracy and correctness by others, and finally edited by one volunteer. The CMCS is not currently used for this joint authorship; it would be desirable to enhance TOOLS or its interfaces to allow this to be done conveniently.

In the absence of a technical note on a topic, the search for information is generally reduced to browsing through the (very long) list of file names and descriptions, looking for likely keywords. Since all human discussions have a tendency to go off on tangents, it is not unusual to find the answer to a question in a place to which simply looking through names and descriptions would not lead the searcher. One of the most asked-for enhancements to TOOLS

is a full-text or keyword search facility, with the typical data base search ("Pascal or Modula with performance and analysis") capabilities. Considering the unstructured nature of the data, and the fluidity of the jargon in the personal computer field, there is evidence in the literature to suggest that a full-text search facility would be more likely to lead to a false sense of security than to do real good.[9]

Styles of usage vary greatly with the method of access.

Along with the problem of there being too much data to search, there is a problem of having too much data with which to keep up. The INFORM and SUBSCRIBE requests for remote users and various front-end programs for users directly linked to IBMPC or its shadows make it possible to read every word that is entered. Obviously, the volume of information available makes this impractical for anyone who has another job to do; the same volume that makes IBMPC a rich source of information makes it a potential source of information overload. Efforts to assess this overload (which is documented only anecdotally at the moment) and address it along various lines (see, for instance, Hiltz and Turroff[10]) are being considered. Without a good feeling for the way the system is actually used, it is hard to choose an overload-reduction strategy that is likely to be effective for a large segment of the community. An informal survey of IBMPC users (conducted by creating an appendable file on IBMPC asking "How do you use the system?") revealed some facts, but a more formal study will be needed before any conclusions can be drawn.

Styles of usage. Styles of usage vary greatly with the method of access. Users with a copy of the data base directly accessible (via LINK) contribute more, seem to be more confident in their use of the system, and have an easier time obtaining answers than do users who access the system via LIST and GET requests, with a (sometimes substantial) time delay between a "read" type request and the arrival of the data. The ability to subscribe to an appendable file hides this difference to some extent; whether or not there is a shadow at the user's location, information added to files subscribed to is available very soon after it enters the data base.

Users also see two distinct types of use of the system: browsing recent activity to keep up with the field, thus possibly coming upon interesting information (the "newsletter" use), and searching the data base for the answer to some specific question (the "search" use). The two types of interaction require very different tools (although there are some tools that would aid in both). Existing front ends seem to serve the newsletter function best; searching is more difficult to do. This is because the system can easily find the most recent items (because it knows the entry times of everything in the system), but it cannot in general find items relevant to a particular subject (since it has no notion of what an item is "about"). One solution to the search problem is to use the pooled memories of the entire user community as a search engine. A user unable to find what he is looking for can add an item to the file called "ISTHERE FORUM," asking if anyone can help out in the search. A typical item in this file might say "I'm pretty sure I saw an account of how to . . . but I can't find it anymore; is there anyone who remembers where this is?" A typical response might simply give the file name, file type, date, and time of the relevant item from someone who happens to remember it (or to remember enough to be able to find it easily).

Sociological aspects. Much has been written on the sociological aspects of CMCSs,[11,12] and no doubt more will be. The sociological data from IBMPC are all anecdotal. We will present a few informal observations here, as possible areas for future work.

- IBMPC has felt the lack of nonverbal communication cues common to most text-based communication. This lack manifests itself as overreaction to joking comments (that would have been accompanied by a disarming smile or a laugh if made in person or through an audio medium), misunderstanding of irony, and confusion as to whether a speaker is serious. The community has adopted some conventions (also used on some other large CMCSs) to make up for this lack. For instance, the symbol ":-)", interpreted as a smiling face turned sideways, is used to bracket light or ironic material. This convention is only as effective as the speakers' perceptions; in many cases, speakers will not use the symbol because they do not realize that there is any danger of misunderstanding.

- Many people present very different personalities through the CMCS than they do in person. In particular, some eloquent and voluminous contributors to the system are quiet and introverted when encountered in person. Despite the fact that TOOLS does not offer the anonymity of pen names or "handles," some people feel much freer to talk electronically than they do in person. Part of this may be due to the fact that the CMCS allows utterances to be proofread and re-examined before being sent, making it easier to be sure of what one is saying.
- The ability to modify or take back items that one has contributed has just recently been added to TOOLS. When it was not available, many users expressed a desire to have it. The combination of easy contribution and (relatively) permanent retention of information raises the danger of a hasty word being on record for a long time. This danger seems to have caused a good deal of anxiety in the user community.
- "Flaming"[13] has been used to describe CMCS users who get carried away by emotion or enthusiasm and enter information into the system that they would probably not have said if the person being addressed were actually present. It has not been a common or a serious problem, however, perhaps because of the more technical subject matter, and the business setting, of the system.[14]
- There seems to be a strong sense of community among the most frequent users of IBMPC. Although many of them have never met (some have probably never been on the same continent), there is a feeling that most of the frequent users know one another and form a reasonably cohesive group that is willing to help new users. At the same time, the amount of bickering among the frequent users is sometimes high, and fewer punches are pulled in discussions between people who know one another comparatively well. The community also does a certain amount of "self-policing;" if someone is overly hostile, threatens to drag some discussion off on a wild tangent, or otherwise has a negative effect on the flow of communication, he or she will generally receive several (generally one-to-one, rather than through TOOLS) messages of admonition. This sense of community is a very positive factor in the success of the system, and many users have cited it as one of the best things about using IBMPC.
- Users have cited many other benefits of the conferencing environment; we will mention a few here. In contrast to what normally goes on during a telephone conversation, a person answering a question can spend a relatively long time finding the answer, without keeping the questioner waiting. Unlike what happens with paper or electronic mail, in conferencing a question asked and answered once can be read by many people in the same situation, rather than being asked many times. In general, this sort of computer conferencing increases the speed, flexibility, and effectiveness of the communication going on. This area deserves further study.

Possibilities

With a user set as large as TOOLS has, there are always some people wanting to use the system in a way that the current implementation does not *quite* allow. Small changes (small in concept, if not in effort required) are made to TOOLS periodically, to more accurately tune it to the users' needs. This section will describe some larger-scale directions in which this sort of conferencing might move.

Some of the future of TOOLS conferencing will depend on the way in which its users and developers think of the system. Although it is very tempting, and sometimes very useful, to draw analogies between components of a CMCS and more traditional parts of the workplace, these analogies can become limits to thinking rather than aids.[15] The analogy of a TOOLS "disk" to a drawer in a filing cabinet might be useful for teaching a new user, but if the developers and users of the system become too comfortable with the analogy, they may not notice desirable enhancements to the system that do not fit in with it. For instance, there are circumstances in which a request to access a given file on a given TOOLS disk might usefully be translated into a request for a file (probably of the same name and type) on another disk, or even under another TOOLS virtual machine. Under the filing-cabinet analogy, this situation would correspond to some file folder in some drawer "really" being another folder in another drawer, or another cabinet. This is not a particularly natural thing to think of doing with real file folders and cabinet drawers, and might not occur to a TOOLS user who was too firmly wedded to the analogy.

Item orientation. Much of the information that TOOLS maintains is at the level of the file. Since, for most conferencing purposes, the most significant entry is the *item* (one component of an appendable file), it might be appropriate to maintain more information at the item level. A convention has emerged to make the first line of each item in the form "Re:"

followed by a statement of the topic of the item. If the TOOLS software itself knew about (and perhaps enforced) this convention, requests such as "fetch all items with the following words in the topic line" or "display the topics of all new items in this file" could be implemented (although such requests are also vulnerable to the earlier comments about the limitations of full-text search). It might also be desirable to mark an item explicitly as a reply to another item, so that a particular conversation thread in a busy file could be easily followed. Given this sort of "item orientation,"[16] front-end programs could take advantage of information about individual items (the fact that the item is a reply, the fact that it refers to another item or file in a certain way, and so on) to provide different views of the data for users with different needs.

Task orientation. Computer-mediated information distribution has the potential to influence many of the tasks performed by its users. In addition to the currently common conferencing and software distribution tasks, TOOLS could be enhanced to support joint authoring, decision-making, progress reports, scheduling of face-to-face meetings, and many other areas. Some of these tasks could be handled with a special user interface to the existing system, or even by a clever use of existing interfaces; others might require modifications to the system in the direction of new entities, new attributes, or new actions.

One of the virtues of TOOLS is that it is very flexible. Because it does not impose a structure on the information entered, it can be used for general information distribution without modification. Whatever structure is added to the system to support specific tasks or to help users filter out information overload, this generality should be preserved.

Conclusion

This paper has presented an example of a large computer conferencing system currently in operation. By increasing the efficiency of technical communications, the system has made thousands of workers more productive and saved many person-hours of duplicated or needless effort. In a very strong sense, this kind of conferencing has changed the way that we do business.

Computer-mediated information distribution will become more important as it penetrates more fields, both in the business and the retail world. The users of the present system cite a large number of benefits, including increased productivity, higher morale, large time savings, and increases in expertise. Computer-mediated communications is a field that deserves a great deal of effort, both in development of new systems and in studies of existing ones.

Acknowledgments

The authors would like to thank all of the people who have made technical contributions to the TOOLS system and IBMPC, including John Alvord, Bob Cronin, Walt Daniels, Barry Dorfman, Rob Golden, David Levine, and Louis Puster, among many others. We would also like to thank Gerald Waldbaum, John Alvord, and Gloria Whittico for their unflagging support of IBMPC at the management and administrative levels, and all of the TOOLS users for their participation and encouragement (especially those who gave us feedback on early drafts of this paper, using the conferencing system).

Appendix A: Authorization levels

Authorization levels control which users may apply which actions to which entities. Each "disk" in a TOOLS system has a set of authorizations; each authorization specifies that some user (or group of users) has some specific power over some file (or group of files). Groups of users may be named by specifying that all users at a given location (regardless of user identification), or all users with a given user identification (regardless of location), be given the authorization. Groups of files may be named by specifying that all files with a given type attribute be included in the authorization. The authorization levels are as follows:

- *ACCESSER:* This "authorization" is in fact negative in effect. A user given ACCESSER authority with respect to some class of files is prevented from executing any actions against those files.
- *GETTER:* Provides read-only access to a group of files; no changes can be made, and no new files can be created. The actions SUMMARY, GET, LIST, QUERY, INFORM, and UNINFORM are the only ones allowed with GETTER authority.
- *APPENDER:* Allows the user to add items to files that he or she owns (or has ADDER authority for). Valid actions for an APPENDER are the same as those for a GETTER, with the addition of the APPEND request. (Note that APPENDERs may not issue OWN or CREATE requests; an APPENDER must be given ownership of some file by a more privileged user before the APPEND action can be used.)

- *ADDER:* Allows adding items to files that are owned by some other user. This authorization is usually restricted to a small set of file types (e.g., FORUM and NOTE) for computer conferencing. The ADDER authority in itself does not authorize access to a disk; it only controls the files that can be appended without the requestor being the owner of the file. A user with ADDER authority thus needs at least APPENDER authority as well if the ADDER is to have any effect.
- *REPLACER:* Allows the user to replace files that he or she already owns, but not to CREATE or OWN any new ones. As for an APPENDER, the user will generally have been given ownership of one or more files by a more privileged user. A REPLACER, then, may, in addition to valid APPENDER actions, REPLACE, HIDE, ERASE, NEWOWN, and REGRESS files that he or she owns.
- *OWNER:* Allows user to OWN and CREATE new files and change file attributes via the SET action (as well as all the actions available to a REPLACER). This is the standard type of user for most applications.
- *PACKAGER:* A packager has the same privileges as an OWNER, except that he may only CREATE or OWN package files. This permits users to be restricted to dealing only with packages. When a user requests ownership of a package file, ownership of all files listed in the package is implicitly requested as well. Thus a PACKAGER may end up with ownership of files of any type, but only if those files are listed in a package file. This helps maintain discipline and makes it possible to determine what package any given file belongs to.
- *PRIV:* This class of user may alter any files on the appropriate disk, whether or not he is the owner of them. The "real" owner of the files changed is warned that the change has been made. A PRIV user may also change authorizations, perform various "dangerous" housekeeping functions (such as erasing old backup copies of files), take actions on behalf of other users, and perform other special actions relating to the disk.
- *SYSTEM:* This class of user may issue certain system maintenance commands and is informed automatically if an error is detected in the operation of the TOOLS system. The SYSTEM authorization is associated with a TOOLS system, rather than with a particular disk. Note that SYSTEM class does not imply any privileges for a particular disk, and so a PRIV, OWNER, REPLACER (etc.) card that covers the user may also be required.

A typical set of authorizations for a TOOLS disk used for computer conferencing might look like this:

```
PRIV VLTVM1 MITCHELL
OWNER * *
ADDER * * ONLY FORUM NOTE IDEAS
ACCESSER KOSSYS2 *
```

This set gives user MITCHELL at location VLTVM1 the PRIV authorization, gives all users everywhere (the asterisk means "any") the power to CREATE and OWN new files, to alter files that they own, and to add new items to files with type FORUM, NOTE, or IDEAS, regardless of ownership. As an exception, no users at location KOSSYS2 are allowed any access to the disk at all.

These authorization levels are the ones built into TOOLS as defaults. A recent experimental enhancement was made to the system to allow owners of individual TOOLS systems to create new authorization levels by specifying exactly which actions users in that class are permitted to perform. For instance, if a TOOLS system is being used for some sort of competitive bidding, an authorization level of "BIDDER" might be created, allowing the users so authorized to APPEND their bids to the bidding files, but not to GET those files (so that previously entered bids could not be examined).

Appendix B: Basic TOOLS actions

There are several classes of actions available to TOOLS users. For the purposes of this list, we will divide them into "read," "write," and "administrative" actions (although this distinction is not actually a part of the system).

The read actions request the TOOLS system to send some piece of information that it holds to the requesting user. They are

- *GET:* Requests that a particular set of files or packages, whose name and type attributes meet given criteria, be sent. The user may specify that only files that have been changed since a certain date be sent and (for files which consist of many items) only items since that date.
- *LIST:* Requests that information about a set of files (specified as for GET) be sent. The list includes the name, type, size, length, last change time, and description of each file. Only files that the requester is authorized to access (through GET) are included in the list.
- *SUMMARY:* Similar to LIST, except that the information returned includes the name, type, owner, and description of each file.

- *QUERY:* Actually a family of actions, the QUERY requests allow a user to find out specific information about a disk, a file, or the TOOLS system itself, or to retrieve a list of the files to which he or she has subscribed.
- *HELP:* Requests system-specific information, set by the maintainer of the particular TOOLS system. This information generally includes (for the system as a whole, and for each disk managed by it) at least the name (and user identification and location) of a person to contact if problems with the system occur.

The write actions are those that actually alter some attribute of some entity in the system. These actions can themselves be usefully split into two classes: those that change some system-interpreted attribute of an entity (such as the list of files that a user subscribes to) and those that actually alter one of the files the system is maintaining. We will cover the first class first.

- *OWN:* This requests the TOOLS machine to register the requesting user as the owner of a file with a given name and type. It does not actually create such a file, however (see CREATE below).
- *NEWOWN:* Changes the ownership information for a given file.
- *SET:* Another family of actions, the SET requests allow the owner of a file to change the name, type, or description attribute of a particular file. One SET request allows a user who is moving to a new location, or getting a new user identification, to inform the system of the change; all occurrences of the old user information are changed to reflect the change.
- *INFORM:* Requests that the user be informed of any changes that occur to a set of files. Optionally, INFORM may be used to "subscribe" to a file by requesting that a copy of every change to the file be sent, rather than just a notification of change. INFORM may also be used to request notification of the creation of new files whose name and type meet the criteria. If the user is itself a TOOLS system, it may request that the actual request that caused the change be forwarded; this can be used to maintain a copy of a subset of the data of one TOOLS system on a remote system (a "partial shadow").
- *UNINFORM:* Cancels the effect of a prior INFORM request.

The next class of write requests actually causes some change to one or more of the files in the TOOLS system. TOOLS is designed with some emphasis on security. Every effort is made to ensure that the person requesting a change to a file is in fact authorized to make that change, and an audit trail is kept of every request.

- *CREATE:* Creates a new file on the disk specified. Except for privileged users, the requestor must either already be registered as the owner of the name and type specified, or the name and type must be unowned. In the latter case, the user is first given ownership, as for an OWN request.
- *REPLACE:* Replaces an already-existing file on the specified disk. Again, the requestor must either be the owner of the file or be a privileged user.
- *REGRESS:* When a REPLACE request is issued against a file, a backup copy of the file is kept. The REGRESS command may be used (by the file owner or by a privileged user) to reinstate the old copy as the current copy (the one sent on GET requests, for instance).
- *APPEND:* Requests that the given information be added as an item to the end of the specified file. This request may be executed by the file owner, by a privileged user, or by a user authorized to add items to files with that type attribute. Another form of APPEND allows a user to modify an item already entered. Only the original contributor of an item (or a privileged user, or the owner of the file) may modify it, and the item is marked with the time and date of the modification.
- *PRUNE:* Requests that any items in the specified file that are older than the specified date be removed. A note is added to the beginning of the file, indicating how many lines were pruned, and by whom.
- *HIDE:* Causes the specified file to become temporarily inaccessible (until the next REGRESS or REPLACE).
- *ERASE:* Erases the specified file (or, if issued for a package, all files referenced only by that package), and deletes all system information concerning it (except for audit trails and the like).

There are various actions, mostly reserved for system administrators and other privileged users, which are used to perform various housecleaning and error-recovery chores. They will not be mentioned in detail here, but it is worth noting in passing that they are all similar in form and content to the usual TOOLS actions and are processed and audited in the same ways. The way in which a system administrator takes care of the system is therefore very much like the way in which a user accesses and contributes to it.

Since most administrators are also users, this means that they have one less thing to learn. It also makes the TOOLS system itself simpler and more consistent.

Cited references and notes

1. The word "file" comes from the operating system on which TOOLS is currently implemented, but the analogy between a TOOLS file and (for instance) a file in a filing cabinet holds quite well.
2. Since the original writing of this paper, a voting facility has in fact been added to TOOLS, but the comments about the flexibility made possible by the user exit points still stand.
3. M. F. Cowlishaw, *The REXX Language*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1985); also, "The design of the REXX language," *IBM Systems Journal* **23,** No. 4, 326–335 (1984).
4. *IBM Virtual Machine/System Product—CMS User's Guide*, SC19-6250, IBM Corporation; available through IBM branch offices.
5. *Virtual Machine/System Product Introduction*, GC19-6200, IBM Corporation; available through IBM branch offices.
6. This is just a very brief overview of the VM environment, hopefully sufficient to make the reader comfortable with the terms that follow; see the references cited for more detailed information.
7. P. Dorn, T. Giblin, and K. Zeliff, *MVS/System Product Release 3 Function and Performance Overview*, GG22-9418, IBM Corporation (January 1981); available through IBM branch offices.
8. *IBM System/38—Introduction*, GC21-7728, IBM Corporation; available through IBM branch offices.
9. D. Blair and M. E. Maron, "An evaluation of retrieval effectiveness for a full-text document-retrieval system," *Communications of the ACM* **28,** No. 3, 289 (March 1985).
10. R. Hiltz and M. Turroff, "Structuring computer-mediated communications systems to avoid information overload," *Communications of the ACM* **28,** No. 7, 680 (July 1985).
11. R. Hiltz and B. Kerr, *Computer-Mediated Communications Systems: Status and Evaluation*, Academic Press, Inc., New York (1982).
12. S. Kiesler, J. Siegel, and T. W. McGuire, "Social psychological aspects of computer-mediated communication," *American Psychologist* **39,** No. 10, 1123–1134 (October 1984).
13. G. Steele, *The Hacker's Dictionary*, Harper & Row, New York (1983).
14. Conferencing has added some other words to the community's vocabulary as well. For instance, "append" is commonly used as a noun, referring to a single item in an appendable file, as in "Did you read that append on debugging?"
15. F. Halasz and T. Moran, "Analogy considered harmful," *Proceedings of Human Factors in Computer Systems* (1982), pp. 383–386.
16. R. A. Flavin, J. D. Williford, and H. Barzilai, "Computer conferencing data structures in the GRANDiose System," *IEEE Transactions on Professional Communication* **PC-29,** No. 1, 34–44 (March 1986).

David M. Chess *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Chess joined IBM in July 1981 at the Research Center. Working at first on VM performance and workload management, he received an Outstanding Technical Achievement Award in 1982 for his contribution to the VM/370 Resource Limiter. When the IBM Personal Computer was announced, he became the first Personal Computer consultant of the Research Center. He then joined the Advanced Workstation Projects Group, working on the prototype that led to the IBM product PC/VM Bond (later receiving a Research Division Award for the work). After a year spent as manager of Advanced Workstation Services, setting up the support structures for workstation users at the Research Center, Mr. Chess now develops ways to enhance the productivity of those users and of the workstation support groups through (among other means) computer conferencing.

Mike Cowlishaw *IBM United Kingdom Limited, Sheridan House, 41–43 Jewry Street, Winchester, Hants SO23 8RY, England.* Mr. Cowlishaw joined the IBM United Kingdom Laboratories Limited at Hursley in 1974, after receiving a B.Sc. in electronic engineering from the University of Birmingham. From then until 1980, he worked on the design of the hardware and software of display test equipment. At the same time, he pursued various aspects of the human–machine interface, including implementation of the Structured Editing Tool (STET, an editor that gives a treelike structure to programs or documentation), several compilers and assemblers, and the REXX programming language. In 1980, Mr. Cowlishaw was assigned to the IBM Thomas J. Watson Research Center to work on a text display system with real-time formatting and on specifications for new facilities for interactive operating systems. He returned in 1981 to the Hursley laboratory, where he completed work on REXX. In 1982, he joined the IBM United Kingdom Scientific Centre to do research on color perception and the modeling of brain mechanisms. He has recently been on secondment to the Oxford University Press, where he was involved with the New Oxford English Dictionary project. He is now in the IBM United Kingdom Research Projects Department. Mr. Cowlishaw has received an IBM Invention Achievement Award and two IBM Outstanding Technical Achievement Awards for the conception, design, development, support, and marketing of the REX (former name of REXX) language and for the development of LEXX, which is a programmable structured editor.

Reprint Order No. G321-5291.