CMU-ITC-86-051

The C-MU PC Server Project

December, 1986

Larry K. Raper

Information Technology Center
Carnegie-Mellon University
CMULKR at PGHVM1
Tie line 363-6775

Unclassified

Unclassified

ABSTRACT

The C-MU Andrew System is a well-known result of a joint study project sponsored by IBM at Carnegie-Mellon University. This paper discusses a related project referred to as the "PC Server". The PC Server permits access to the Andrew network, and a subset of its services, from the entire family of IBM personal computers.

The PC Server provides a mapping of the PC-DOS file system onto the Andrew network's VICE file system, such that existing PC software can transparently access files using the campus network. This extends to PC users the advantages of a large, centrally managed, shared file system that is functionally indistinguishable from an ordinary local disk drive. The PC Server also provides peripheral capabilities for synchronizing the time in the local machine with the campus network, changing account passwords, and accessing the Andrew printing, mail, and bulletin board subsystems.

This paper discusses the rationale behind the PC Server and describes the organization and structure of its major components, as well as the interfaces and protocols used among them.

ACKNOWLEDGEMENTS

The project described in this paper is the work of the author, with invaluable assistance from the following people.

Nathaniel Borenstein	for CUI, MS, and many suggestions for SNAP.
Mark Chance	for enhancing CUI and porting it to the IBM PC environment.
John Leong	for providing user documentation, expert guidance, and unfailing enthusiasm.
Mark Lorence	for device drivers for the IBM token ring.
Drew Perkins	for the Microsoft C implementation of MIT's PC/IP.
Jon Rosenberg	for the GUARDIAN, and help with the design of SNAP.
Bob Sidebotham	for his early help with a prototype server.
F. Don Smith	for advice and help with IBM token ring problems.

Special thanks go to Paul Crumley, John Howard, Bob Kubiak, Jonathan Miller, Barry Silverman, and Bryan Striemer for daring to be early users of the PC Server software and supplying valuable feedback. Thanks also to Mike West and Mahadev Satyanarayanan for their support and advocacy of the intermediate server approach for access to the VICE file servers.

Unclassified

CONTENTS

Unclassified

INTRODUCTION

In 1982 IBM and Carnegie-Mellon University began a large scale joint study project to explore the design and implementation of a campus-wide network of computer workstations. The goal of the campus network was to permit university students, faculty, and staff to function as an extended community, having many of the common bonds of a traditional time-sharing system, but without the accompanying limitations on system capacity and user community size.

To address the problems inherent in the project and oversee the orderly development of solutions, IBM and C-MU established the Information Technology Center (ITC). The ITC is housed in the University Computing Center and staffed with full-time IBM and C-MU employees working together. An IBM site manager oversees the local IBM personnel and reports directly to IBM's Academic Information Systems (ACIS) development organization in Milford, Connecticut, while also reporting to the C-MU appointed ITC Director. The ITC's mission was to design the network, oversee its implementation, and provide whatever software would be needed.

The C-MU Andrew System is the principal result of this effort. Andrew consists of a graphically oriented UNIX-based[1] environment (called VIRTUE) built on a high performance LAN-based distributed file system (called VICE). The VIRTUE software runs on a variety of engineering/scientific workstations that feature a mouse and a large APA display, including the IBM RT. The VICE software is distributed among VIRTUE workstations and special file server machines that support the attachment of large amounts of DASD. Interconnecting the many physical elements of the Andrew system is a series of local area networks (principally using IBM token ring or Xerox Ethernet technology) that are joined by routers to a backbone network. The packet format supported by the routers is the Department of Defense standard Internet Protocol (IP); the same protocol used on the ARPANET and by many research universities.

Work on the various components of the Andrew network has taken the ITC well over three years. Andrew is currently in widespread use throughout C-MU, and all incoming freshmen receive Andrew accounts. The network was officially dedicated in November, 1986, and IBM (ACIS) has declared its intention to provide a product based on Andrew for the general university market. The ITC continues to be involved with refinements to the Andrew software and the expanding use of the network within the university.

Andrew, however, presents only a partial solution to the original goals of the joint study. Providing a network of computer workstations that spans an entire university campus and serves users of many disciplines and skill levels requires an acknowledgement that diverse machines and applications already exist there.

[1] UNIX is a trademark of AT&T Bell Laboratories. There are numerous versions. In this paper UNIX always refers to Berkeley UNIX version 4.2. On the IBM RT this corresponds to IBM's ACIS 4.2 operating system, and should not not be confused with IBM's AIX, a derivative of UNIX System V.

It is seldom the case that users will simply abandon their existing computer facilities to take on expensive, new, untried, systems – no matter how appealing the new capabilities may seem. Consequently, now and for the foreseeable future, the campus network needs to accommodate the attachment of many different machines and network media.

The PC Server software described in this paper addresses the general problem of how to attach non-Andrew workstations to the campus network for a specific set of machines, namely, the IBM PC family. The IBM PC was selected as a particularly important attachment case, due to its already pervasive presence on campus and the enormous body of existing IBM PC application software.

OVERVIEW

The C-MU PC Server provides users of IBM PCs with access to three distinct types of services that are normally available to direct users of the Andrew system. These are:

1. File system services.

2. Printing services.

3. Mail and bulletin board services.

These services were chosen for initial implementation on the IBM PC machine family because they address expressed needs of current users of these systems and do not necessarily presume the availability of specialized graphics input or output devices (such as the mouse and APA display found on all of the "advanced function" workstations supported by Andrew). This second reason is of some importance when one considers the variability of IBM PC configurations, the widespread use of text-only displays, and the need to allow users with existing systems to connect to the campus network with only a minimal cost outlay for additional hardware.[2]

For any service provided over the network for an IBM PC, there is at least one Andrew advanced function workstation involved in the role of a server or host machine. The PC and its server machine maintain one or more sessions for the duration of time that service is to be provided. Sessions are established by a special PC program called LOGIN and last until a long period of inactivity causes the server to terminate automatically, or the PC user issues an explicit LOGOUT. The LOGIN/LOGOUT process permits users to identify themselves and allows the Andrew network to validate their right to access resources.

Although any Andrew workstation can potentially be used as a PC Server host, the university has chosen to provide dedicated machines for use as public PC Servers. Each PC Server machine can support between 15 and 20 PC users simultaneously.

FILE SYSTEM SERVICES

Think of the Andrew VICE file system as an enormous, global, UNIX file system, accessible to all users anywhere on the network. With the PC Server software installed, a PC DOS user is able to manipulate the VICE file system, as though it consisted of additional instances of the local PC DOS file system. Each such

[2] The PC Server software requires only the addition of a network adapter and 100K to 140K of dedicated PC memory, depending on the type of adapter. All versions of PC-DOS from 2.0 through 3.2 are supported.

network instance of the PC-DOS file system is assigned a unique drive designator, with the root directory of that drive corresponding to a user-selected VICE directory. That is, to a PC user, discrete portions (subtrees) of the VICE file system appear to be "network drives" where PC DOS files and subdirectories of files are stored.

In general, all PC-DOS file system operations are translated to corresponding actions in the VICE file system, so that from a PC the full functionality of PC-DOS can be apnlied directly to VICE. This type of relationship is sometimes described as an injective mapping, where all PC-DOS operations have a counterpart in the host file system, but the inverse is not true. Some VICE operations have no equivalent PC-DOS actions.

Unlike "store and forward" or file transfer systems (such as IBM's VNET and SNADS, or the DARPA FTP utility), network drives permit a type of "hands on" access that occurs in real time and allows the end user to manipulate the actual host system. When a PC-DOS user (operating on a network drive) makes a new directory, erases or renames a file, or simply updates a file with new data, the VICE file system itself is changed, and direct users of the UNIX-based Andrew system would see the same effect. Except for the use of additional drive designators, network drives are functionally indistinguishable from local disk drives, differing primarily in speed and capacity.

Since the PC Server presents the illusion of VICE as a transparent extension of PC-DOS, arbitrary subtrees in VICE may be designated as network drives and ordinary commercial PC Software packages can be used directly to store, update, and retrieve files kept in VICE. But despite the fact that the VICE file system is ostensibly similar to a normal PC-DOS file system, it offers some important advantages for PC-DOS users.

- Because the VICE file system is centrally managed by a professional computer staff, PC files that are kept in VICE are automatically backed up by regular VICE backup and archiving mechanisms.

- Files kept in VICE are accessible from other workstations (or other PCs) attached to VICE anywhere on the network.

- Files may be kept in private or shared directories.

- The amount of space available for the storage of files is restricted only by the quota limitations associated with the user's Andrew account.

- With some restrictions, text files may be used from both PC-DOS and Andrew (i.e, UNIX) workstations.

These advantages are offset somewhat by the fact that the time to read a file kept on a network drive is roughly comparable to the time it would take to read the same file from a diskette. But for users willing to trade fixed disk performance for diskette-like performance, or for users without a fixed disk, a network drive can make an attractive repository for PC files.

PC-DOS users tend to find the concept of a network drive a relatively natural extension of their environment. Each reference to a PC-DOS file already includes either an explicit or implicit drive identifier.[3] PC-DOS also comes with a device driver that permits a user to dedicate an area of memory to be used as a high speed "virtual disk." The "virtual disk" is similarly distinguished from an actual disk by the assignment of a unique drive designator. Thus the use of new drive designators to specify independent instances of PC-DOS file systems is a standard mechanism, and an obvious basis on which to redirect file system references.

TRANSLATED VERSUS IMBEDDED FILE SYSTEMS

A PC Server network drive differs significantly from a virtual disk. Besides being noticeably slower, it does not contain a PC-DOS partition record, BIOS Parameter Block (BPB), or File Allocation Table (FAT). The reason is that PC-DOS is not managing the implementation of network drives and internal DOS constructs are not appropriate. The underlying functions of space management and device management are actually performed by the host file system (VICE), and are completely transparent to PC-DOS.

The PC Server software monitors all requests for PC-DOS services and diverts those that are intended for network drives. This interception occurs at the file system interface rather than the device driver[4] level. With the exception of PC-DOS utility functions that are actually implemented as application programs (themselves users of the file system interface), all PC-DOS components are sheltered from any knowledge of the existence of network drives.

This is an important point because the PC Server software is able to maintain the meaning of a file system operation by translating the high level abstractions of the PC-DOS file system (drives, directories, and files) directly into corresponding notions in the VICE file system. It is this forced correspondence between the two systems that makes the actions of PC programs on what are presumably PC-DOS files visible as equivalent operations on the same files to normal VICE users, and vice versa.

At the file system interface, all the application-level semantics of a given operation are well-defined and fully known. By the time PC-DOS passes the request to a device driver its meaning has been reduced to underlying physical operations of the form "read or write x blocks of data starting at block number

[3] One drive is always designated as "current" and file names that do not explicitly identify a drive are, by default, associated with the current drive.

[4] A device driver is an architected means of adding device support to PC-DOS. There are two types of device drivers, "block" and "character." A block device driver supports a device that is capable of housing a PC-DOS file system. Character device drivers support all other types of devices.

y" and the higher level request that was originally expressed in terms of files or directories is no longer discernible. The difference in these interfaces is illustrated in Figure 1 on page 7.

It would have been a considerably easier task to implement a PC Server connection at the device driver level, since there are fewer, and more primitive, services provided through this interface. The resulting system, however, would be significantly different. The blocks or space of the host file system could be used to hold the PC user's data, but these would have no meaningful relationship to user—level objects in the host system (like files and directories). Although imbedded within the host system, all of the data and its interrelationships would be privately managed by means of remote PC—DOS operations.[5]

With this approach, the host system (VICE) would view the entire PC—DOS file system as a single file of unknown internal structure. PC—DOS files and directories would not be accessible to normal VICE users except through the use of specially constructed utility programs that understood the internal organization of a PC—DOS file system and applied that interpretation to the data in the VICE file. Similarly, PC—DOS users would not have direct access to normal VICE files, but would require new and overt import/export mechanisms.

Since the PC Server software support applies to the file system interface, a small number of special purpose DOS commands will not operate on network drives. Among these are CHKDSK, DISKCOMP, DISKCOPY, FDISK, FORMAT, RECOVER, and SYS. Typically these programs perform device—specific functions that require them to access drives using lower level system interfaces. If applied by accident to a network drive, they produce harmless error messages and terminate.

DIFFERENCES BETWEEN PC—DOS AND VICE

Because the original design of the PC—DOS file system was influenced by the prior existence and widespread use of UNIX, and because the VICE file system preserves most of the semantics of a true UNIX file system, there are many similarities to begin with.

- Both systems have a hierarchical directory structure.

[5] Two examples of a PC—DOS file system imbedded within a different host file system are IBM's PCVMBOND product (PC Bond - P/N 5664—298, VM Bond - P/N 6476128) and the RVDs (remote virtual disks) of MIT's Project Athena.

The IBM PC Local Area Network Program (P/N 6280083), on the other hand, provides another example of a network drive implemented at the file system interface. An extension of PC—DOS called "the Redirector" traps file system calls and transfers them to a host program called "the Receiver". In this case, the host file system is a standard PC—DOS file system maintained on a remote machine, and the translation between client and host file systems is straightforward.

```
   Programming Operations     Abstractions

   Application Program
     Open/Close ------+       Drives
     Read/Write       |       Files
     Rename/Erase     |       Directories
     Mkdir/Rmdir      |       Offset addressing
     Directory lookup |
                      V                          Intercept here
   ------ File System Interface (INT 21h) -----> to translate to
                                                 a foreign file
   PC-DOS                     Devices            system.
     Read/Write -------+      Clusters (FAT)
     x blocks at y     |      Logical block addressing
                       |
                       |
                       V                         Intercept here
   --------- Request Header Interface ---------> to imbed within
                                                 a foreign system.
   Block Device Driver        I/O Adapters
     BIOS functions           Sectors
                              Geometric addressing
```

Figure 1. PC-DOS Interface Levels and Available Abstractions.

- Both systems allow arbitrary nesting of directories, and construct path names for files as a succession of nested directory names separated by a single character.

- Both systems use . and .. to refer to the current directory and the parent directory, respectively.

- Both systems provide only a basic "byte offset" access method. Higher level access methods must be implemented as application subroutine libraries, with operations ultimately expressed in terms of the basic functions. This keeps the number of formal file system operations small and manageable.

More interesting, perhaps, are the ways that the two file systems differ.

- VICE file names, like UNIX, are case-sensitive; PC-DOS file names are not. That is, **File-X** and **file-x** refer to two different VICE files, but would be considered the name of the same file by PC-DOS.

- PC-DOS directory and file names are limited to be at most 8 characters in length with an optional 3 character suffix, separated by the first portion of the name with a period (.). VICE file names are allowed to be much longer, with imbedded periods having no particular significance.

- UNIX files have an "owner" based on the userid of the file's creator. The owner of the file is allowed to specify whether other users can access the file, and the types of access that will be permitted. The VICE file system differs slightly from standard UNIX in this respect. In VICE, access information is maintained for directories rather than individual files.

Both VICE and UNIX support a richer set of access rights than PC-DOS, which also lacks the concept of a file owner.

- PC-DOS and VICE have slightly different conventions about line delimiters in text files (PC-DOS uses ASCII 13,10 - carriage return, line feed as a line delimiter, while VICE/UNIX uses a single line feed).
- PC-DOS files may be marked as hidden, so that they will not appear in directory listings. In VICE and UNIX a similar effect is provided for files whose names begin with a period (.).
- VICE and UNIX use a forward slash (/) as a separator for path name elements, while PC-DOS uses a backward slash (\).[6]
- VICE and UNIX both support certain operations that have no corresponding features in the PC-DOS file system, such as physical or logical links, and a permission flag for "execute" access.

Some of these differences can be compensated for in obvious ways (or at least ways that are transparent to a user), but others cannot. Those features of VICE not present in PC-DOS are unsupported for network drives. Access to files or directories in VICE that are restricted based on userid, is governed by the information supplied by the PC user during the login process.

The two aspects of VICE that are most difficult to conceal are long file names and the difference in text file formats. Neither of these differences are likely to be noticed by PC users that use the PC Server software primarily as a means of keeping their PC-DOS files in the VICE file system. But another class of users, those who need to access VICE files from both PCs and VIRTUE workstations will be aware of the differences.

For this second class of users the PC Server package includes the LS and CP commands. These commands are PC programs that are modeled after UNIX commands with the same names. They have the same syntax as their UNIX counterparts, perform the same functions, and produce approximately the same output.

LS is the UNIX analog to the PC-DOS DIR command, but has many more output options and formats. When applied to a network drive from an IBM PC, LS will display the same results that a VIRTUE user would see on a UNIX workstation, including the real VICE file names, regardless of their length or case.

CP is the UNIX equivalent of the PC-DOS COPY command, but will accept target or source file names (for files on network drives) without the name restrictions

[6] Although PC-DOS uses the backward slash as a separator for external interactions with users, all of the internal programming interfaces accept forward or backward slashes interchangeably.

imposed by PC-DOS. In addition, if CP determines that you have requested it to copy a text file from a PC-DOS drive to a network drive (or the other way around), it will convert the text formats automatically. By using LS in conjunction with CP, a PC user can determine the true name of a VICE file and create a copy with a legal, and meaningfully chosen, PC-DOS file name.

UTILITY FUNCTIONS

For convenience, several utility functions are included in the PC Server package. A SETTIME command synchronizes the date and time information in the PC with that maintained by the Andrew network. An ST (status) command reports the version numbers of the installed PC Server software components and presents summary information about network drives. A UNIX-style PASSWD program permits the PC user to change the password associated with his Andrew account periodically.

Another utility program is currently under development that will allow PC users to inspect and modify the default access control information associated with their VICE directories (all Andrew accounts are automatically provided with both public and private VICE directories). And the list of utility programs is likely to grow further in the future.

PRINTING SERVICES

A wide variety of quality printers are available to users of the Andrew network, including IBM 3820, IBM 3812, and IBM 6670 printers. Some printers serve particular campus organizations, while others are available for general use.

In the PC Server environment, a special application program (PR) provides PC users with the means to ship a file to any of the network printers. The PR command presents panels that permit the specification of file name, printer, delivery information, fonts, print styles, paper orientation (portrait or landscape), duplex or simplex pagination, and detailed page layout information. A batch mode of operation permits the PR command to be used in PC-DOS BAT files, bypassing the interactive panel mode.

The PR command is simply a remote interface to the Andrew printing subsystem, which queues the file for eventual output on the appropriate printer.

MAIL AND BULLETIN BOARD SERVICES

One of the most successful ways of building a sense of community among a large scattered group of computer users is to provide an effective, reliable, and easy-to-use mail and bulletin board system. Next to the VICE file system, the Andrew mail and bulletin board programs are among the most heavily used, and generally the first to be mastered by novice users.

Both private mail libraries and public or private bulletin boards have a common underlying structure that is understood and maintained by a special Andrew system component referred to as the "message server". Users may interact with the message server using either of two interface facilities. One has a graphic orientation and relies heavily on pointing operations performed with a mouse; this interface is for direct users of VIRTUE workstations. The other is called CUI (for Common User Interface) and is a command-oriented interface that was specifically designed so that it could be used from UNIX shell scripts, dial-up ASCII terminals, VIRTUE workstations, and be easily ported to other computer systems.

The PC implementation of the CUI provides PC users with all of the same functions available on a VIRTUE workstation, including the ability to compose, send, and receive mail, review pending or old mail, and read, create, subscribe, or contribute to any of the Andrew bulletin boards.

PC SERVER ORGANIZATION AND STRUCTURE

The PC Server consists of a suite of related programs, some of which run on IBM PCs, XTs, or ATs, and others that reside and execute within the Andrew environment.

In general, the division of function between the PC portion and the Andrew portion exhibits the same characteristics commonly seen in an application program and a subroutine library. The purpose of the subroutine library is to provide a convenient set of related services for many potential application programs, while hiding the underlying complexities of their implementation. In a distributed system such as this, the dividing line between application and subroutine involves the insertion of a synchronizing communications protocol. The services (subroutines) reside at a remote site, appropriate for access to the resources they manipulate. The application deals directly with the end-user, accepting input and producing the expected outputs, while invoking the services it needs in the form of remote procedure calls.

The major components of the PC Server system are depicted in Figure 2 on page 12. The large box at the top of the figure illustrates the functional divisions found within an IBM PC. The three smaller boxes shown beneath it, indicate the related service components that execute in a UNIX workstation somewhere in the Andrew network. These depict instances of three different processes, all of which could normally operate within a single machine.

The MS (Message Server) process is the standard Andrew component for storing and retrieving the message entries that comprise the mail libraries and bulletin board system. The elongated box labelled VICE represents the many service machines that support the distributed VICE file system. A brief summary of the other components is given in this section. Those that are unique to the PC Server environment itself and play no other role in the Andrew system are discussed in greater detail. To begin with, we turn our attention to the thread that binds the pieces together.

SNAP

SNAP (for Simple Network Access Protocol) is the name of the private communications protocol used by the major components of the PC Server software. It is an IP-based protocol that was specifically designed to provide the minimum set of capabilities needed to implement a robust remote procedure call facility. It provides a secure, reliable, sequenced, half duplex, low overhead connection (referred to as a conversation) between a client and a server process.

SNAP is an asymmetric protocol, just as the procedure call relationship between program and subroutine is asymmetric. The server process (subroutine) is presumed to be completely passive, responding only when invoked by a client (program). Messages of arbitrary (but predeclared maximum) length may be sent from

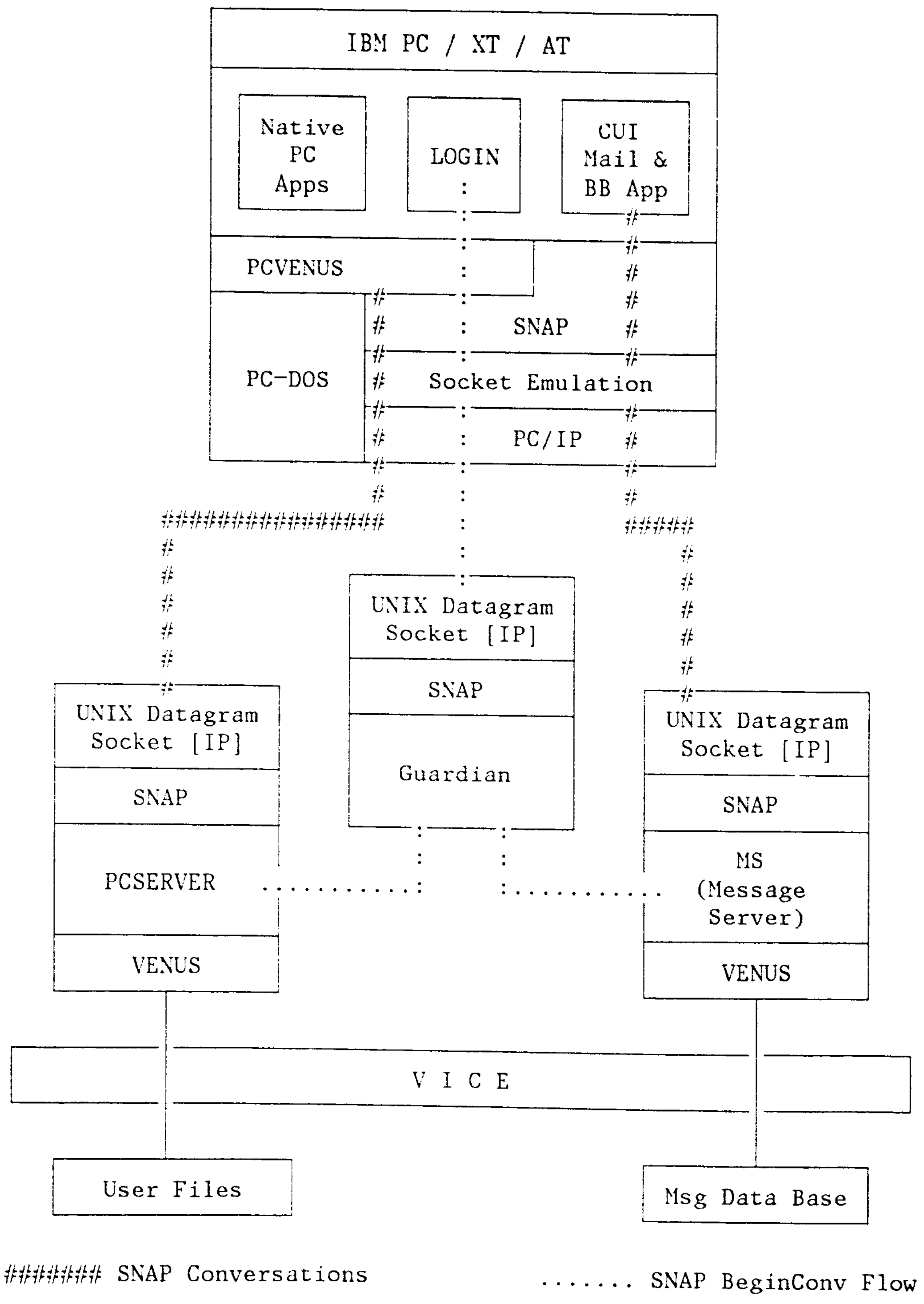

Figure 2. PC Server Block Diagram

the client to the server, and received as a reply. Multiple conversations may exist between one client and one server, one client and many servers, or one server and many clients. A process may be both a SNAP client and a SNAP server, but cannot be in conversation with itself.

SNAP services fall into three distinct categories: client, server, and data exchange services. All SNAP services are synchronous; the invoking program relinquishes control until they complete or time out.

SNAP CLIENT SERVICES

SNAP client services are employed at the user (i.e, main program) end of a distributed application.

SNAP_ClientInit must be requested once by each program prior to any other use of a SNAP client service. This permits the SNAP protocol to perform any global client initialization procedures. The external effects of this operation are unspecified.

SNAP_BeginConv results in the construction of a conversation between the client and server processes, and may be performed multiple times. A client may have several conversations simultaneously, all with the same server or different servers.

In order to establish a conversation SNAP consults with a special authentication component on the server machine referred to as the GUARDIAN. The GUARDIAN, although not part of the SNAP protocol per se, is involved in a three-way exchange of information with the client and server processes. This interaction is illustrated in Figure 3 on page 14.

If the GUARDIAN can successfully authenticate the client process, it locates or creates an appropriate instance of the server on the client's behalf and returns the server's network address. The GUARDIAN also supplies both the client and server processes with a 48-byte key, that can be used (at the client's option) to encrypt all subsequent conversation traffic.

To verify that the server process is ready to enter into a conversation an initial negotiation occurs between the client and server regarding packet sizes, acknowledgement windows, and other internal session parameters. Upon the successful completion of these exchanges, a conversation exists between the client and server processes.

SNAP_SendWithReply provides the basic mechanism for a remote procedure call. A message of arbitrary length containing an uninterpreted stream of

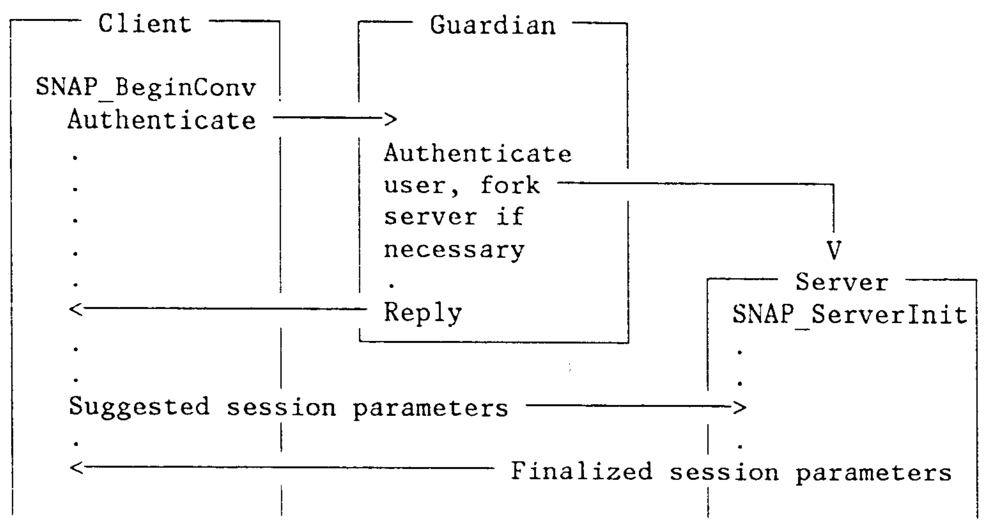
Client
Guardian
SNAP_BeginConv
Authenticate
Authenticate user, fork server if necessary
Reply
V
Server
SNAP_ServerInit
Suggested session parameters
Finalized session parameters

Figure 3. Three-way Exchange during SNAP Begin Conversation

bytes is sent over a specified conversation and an answering message (or reply) is returned. The reply is also an implicit acknowledgment that the server program has received and processed the message.

SNAP_SendNoReply performs an unacknowledged transfer of a message over a specified conversation. A series of one-way messages sent in this fashion can be acknowledged by an application-level convention that the last one would be sent using SNAP_SendWithReply and include summary information for the detection of missing messages. The subsequent reply would acknowledge all messages up to that point or indicate those that were not received.

SNAP_SetConvParms allows a client program to alter the encryption option or time-out parameters associated with a specified conversation.

SNAP_EndConv terminates the specified conversation.

SNAP_ClientTerm must be used once prior to program termination. It automatically performs SNAP_EndConv processing for any conversations still in progress and undoes the effects of SNAP_ClientInit.

The general format of a client program is illustrated by the following piece of C pseudo-code (arguments to SNAP functions are omitted for the sake of simplicity). In this example, the mainline program uses the services of the subroutine library as though they were available locally. In actuality only small stubs, containing SNAP_SendWithReply calls are linked with the program.

```
main() {
   SNAP_ClientInit();       /* Initialize SNAP support    */

   /* Interact with user to obtain required inputs.       */
   /* Perform the mainline application logic.             */
   /* Where needed manipulate remote resources using      */
   /* appropriate service routines.                       */

   SNAP_BeginConv ();       /* Establish a conversation   */
   result1 = service_1 (arguments);
   result2 = service_2 (arguments);
   result3 = service_3 (arguments);
   SNAP_EndConv ();         /* Terminate the conversation */

   /* Produce output for end user                         */

   SNAP_ClientTerm();       /* Terminate SNAP support     */
   }

service_1 (arguments) { /* Stub for service_1 subroutine. */
   op_code = 1;         /* construct message with op_code */
   make_message (op_code, arguments);    /* and arguments */
   SNAP_SendWithReply (message, reply);
   return (reply);
   }

service_2 (arguments) { /* Stub for service_2 subroutine. */
   op_code = 2;         /* construct message with op_code */
   make_message (op_code, arguments);    /* and arguments */
   SNAP_SendWithReply (message, reply);
   return (reply);
   }

 Etc.
```

SNAP SERVER SERVICES

These services are provided for use by server programs. They support the semantics of message handling at the subroutine end of a remote procedure call.

SNAP_ServerInit initializes SNAP for server related processing and is required prior to the use of other SNAP server functions.

SNAP_Accept waits until the next message has been received from any conversation. An optional time-out parameter may be specified to avoid an indefinite wait. SNAP either returns with the message, an indication of whether a reply is needed, and a token that identifies the associated conversation, or indicates that the optional time-out interval has expired.

SNAP_Reply sends an answering message back over the specified conversation. Unsolicited replies are not permitted.

SNAP_MsgPending can be used to determine whether the next SNAP_Accept call will wait or return immediately with a message.

SNAP_ConvCount indicates the number of conversations in progress with all clients.

SNAP_ServerTerm forcibly shuts down all current conversations and undoes the effects of SNAP_ServerInit.

The following piece of C pseudo-code illustrates the typical structure of a SNAP server program (again, most arguments are omitted for simplicity). At this end of a conversation the actual subroutines that perform the indicated services would be linked into the server program.

```
main() {
   SNAP_ServerInit();           /* Initialize SNAP support. */

   do {                         /* Process each message by  */
      SNAP_Accept();            /* performing the requested */
      Extract op_code from message;         /* service. */
      switch (op_code) {
         case 1: result = service_1(arguments_from_message);
                 break;
         case 2: result = service_2(arguments_from_message);
                 break;
         case 3: result = service_3(arguments_from_message);
                 break;
         ...
         default: result = no_such_service;
         }
      If (reply_needed)         /* Send the results back to */
         SNAP_Reply (result);               /* the client. */
      } while (SNAP_ConvCount() != 0)
                                /* Terminate when all       */
   SNAP_ServerTerm();           /* conversations have ended */
   }

service_1(arguments)
   {  /* Actual subroutine code for service_1 goes here */ }

service_2(arguments)
   {  /* Actual subroutine code for service_2 goes here */ }

service_3(arguments)
   {  /* Actual subroutine code for service_3 goes here */ }

Etc.
```

It should be clear from these (admittedly sketchy) examples that an integrated application can be produced from a distributed application by removing all the SNAP calls, discarding all of the mainline code from the server and the stubs supplied with the client, and linking the remaining portions into a single program. The inverse is also true. With some care, a distributed application (that can be invoked from machines removed from the program's actual resources) can be created from a conventional application, by segregating all subroutines that manipulate local resources, supplying stubs at the client end, adding the main superstructure shown above for the server and some additional SNAP calls at each end. The CUI and MS (Message Server) components were developed in this fashion from a single, integral program.

SNAP DATA EXCHANGE SERVICES

None of the SNAP client or server operations are sensitive to the content of the data they transmit over the network. To accommodate the fact that the network is comprised of different kinds of computers, which may (and do) use differing data representations, the SNAP protocol provides two policies for the exchange of data.

1. All character data will be transmitted using 8-bit ASCII code and all binary data as 32 bit integers with the bits arranged in a well-defined ordering referred to as "network byte order".

2. All other types of data must either be coerced to one of the above types or encapsulated as a sequence of uninterpreted 8-bit bytes with end-to-end understanding of their representation.

These policies are expected to be observed voluntarily and are not enforced in any way by SNAP. SNAP does, however, offer the following services (for both client and server programs) that are designed to facilitate the data exchange policies. All of the PC Server components use these services when packing remote procedure call arguments into a message and unpacking the values returned in the replying message.

SNAP_AppendIntToMsg places a 32 bit integer into a message buffer in network byte order. Binary values of other lengths must first be coerced to 32-bit format.

SNAP_AppendStringToMsg transforms a zero-terminated character string to 8-bit ASCII[7] and places it into a message buffer in an encapsulated form.

SNAP_AppendBytesToMsg places a specified number of uninterpreted 8-bit bytes into a message buffer in an encapsulated form.

---

[7] At the present time this is a null transformation for all machines on the Andrew network.

SNAP_ExtractIntFromMsg converts a 32-bit network byte order binary value from a message buffer to a 32-bit integer with bit ordering appropriate to the current machine.

SNAP_ExtractStringFromMsg returns a zero terminated character string from a message buffer, transformed[7] to the native character code of the current machine.

SNAP_ExtractBytesFromMsg returns an uninterpreted sequence of 8-bit bytes and their length from a message buffer.

The subroutine stubs shown previously with the client pseudo-code would use these functions to marshal actual arguments into a message and extract the results from the reply.

PORTABILITY CONSIDERATIONS

A communications protocol needs an implementation on every type of machine that is to use it, consequently portability was an important consideration in the design of SNAP.

SNAP is written entirely in C, a language for which complete implementations exist for most computer systems. C is also known as a low-level language whose operations correspond well with the hardware capabilities of a broad class of machines, and one that requires only the most primitive of run-time environments. This makes it an ideal implementation language for many system functions.

SNAP also makes only minimal assumptions about the capabilities of the operating system beneath it. All of its actions are ultimately expressed in terms of the following system functions (which are native to Berkeley UNIX, but easily emulated on an IBM PC and other systems):

socket	Creates a port for the sending and receiving of datagram packets.
select	Waits for the arrival of a datagram packet subject to a time-out.
sendto	Sends a datagram packet.
recvfrom	Reads a datagram packet from a socket.
time	Returns the elapsed time in seconds since a fixed reference point.
gethostname	Returns the symbolic name of the current machine.
gethostbyname	Translates a symbolic machine name to an IP station address.

getservbyname

Translates the name of a service to an IP port number.

These system functions define a specific interface to an implied, underlying IP packet delivery mechanism.

For the IBM PC environment the MIT PC/IP package, also written in C, was used as the basis for this capability. A small layer of code shown in the PC Server Block Diagram as "Socket Emulation" provides the specific interfaces needed for the above services. These pieces are combined with the SNAP protocol layer to form a single PC-DOS "terminate and stay resident" program. Once executed it becomes an extension of the user's operating environment. The SNAP services are exposed for general use as software interrupt functions, with language bindings currently implemented for both C and Assembler programs.

GUARDIAN

The GUARDIAN is a daemon process that is started on all VIRTUE workstations during their boot sequence, and a standard component of the Andrew system. It performs authentication functions on behalf of programs that need to establish communication sessions with server processes. In addition, it oversees the operation of server processes, creating them as needed, and notifying clients when they terminate. It is the GUARDIAN that determines whether or not a particular workstation may be used as a host machine for remote SNAP conversations, and grants or denies access accordingly.

All PC Server users are required to have an Andrew account. When the GUARDIAN creates an instance of the PCSERVER or MS (Message Server) programs in response to a PC user's login request, these processes have the same capability to access system resources as the user would have when logged into an Andrew workstation directly. They become, in effect, agents that act as surrogate users on behalf of particular PC clients.

VENUS

VENUS is the Andrew component in every VIRTUE workstation that understands the VICE file system. It makes the interactions with VICE transparent to the other Andrew system components, and gives the impression that each workstation has access to a large shared UNIX file system. Special hooks in the UNIX kernel allow VENUS to trap and inspect all file system calls. For efficient operation VENUS manages a sizeable local file cache, so that commonly used files need not be transmitted to or from the remote VICE file servers unless they have been modified.

PCSERVER

PCSERVER offers a collection of generic file system services and three special operations that may be accessed as remote procedure calls using the SNAP protocol. When executed, these services operate on the UNIX file system of whatever machine is hosting the PCSERVER process. In the Andrew environment, the VENUS component of the local machine causes the operations to be applied to the VICE file system. Access to VICE from an IBM PC is thus an indirect process, always going first through the PCSERVER, rather than dealing directly with VICE file servers.

This approach was chosen for several reasons.

1. Initially, it allowed the development of the PC Server software to be decoupled from concurrent VICE development work.

2. It allowed the specification of interfaces that were more appropriate for PC-DOS use and cognizant of the limitations of the IBM PC family. As an example, VICE supports only whole file transfers, whereas PCSERVER performs partial file transfers.

3. It avoided the need for full duplication of function between VENUS and PCVENUS (with the current design all file caching is performed only by VENUS).

4. It permits the PC Server software to be hosted by a standard UNIX system, without requiring the use of the VICE file system at all.

   Of course, for PC users, this type of configuration lacks many of the advantages that come from the use of VICE (such as global access to the same file from any workstation in the network without regard to any particular server machine). Still, it can offer access to larger DASD devices than can be attached to IBM PCs, and also provide effective sharing mechanisms among users of the same host file system.

The PCSERVER program provides the following file system services for use as remote procedure calls. All date or time information is provided or returned in standard PC-DOS formats.

PCS_Open opens a file for input, output, or unspecified access, given a fully qualified filename, and returns a handle for subsequent access, along with file mode, date, time, and size information. Special heuristics are applied to the filename to compensate for the lack of case-sensitivity by PC-DOS.

PCS_Close closes the file associated with a given handle.

PCS_Read obtains data from an open file, given a handle, starting offset, and length. A return length value indicates the number of bytes actually read.

PCS_Write places data into an open file, given a handle, and starting offset. If the length of the data supplied is zero, the file is truncated or extended to the size indicated by the starting offset value.

PCS_DirSearch obtains information about a file, given a fully qualified pathname, a generic filename with optional pattern matching characters, and a starting filename (for iterative invocations). It returns the name of the first matching file, along with mode, date, time, and size information.

PCS_RemoveFiles erases one or more files, given a fully qualified pathname, and a generic filename with optional pattern matching characters.

PCS_RenameFiles changes the name of one or more files, given a fully qualified old pathname, an old generic filename with optional pattern matching characters, a fully qualified new pathname, and a new generic filename with optional pattern.

PCS_MkDir Creates a new directory given a fully qualified pathname.

PCS_RmDir Removes an empty directory given a fully qualified pathname.

PCS_ChMod Changes the mode properties of a file, given a fully qualified filename.

PCS_SpaceQuery returns space accounting information from the host file system.

PCS_TimeStamp sets a file's last updated time and date information, given a file handle, and appropriate time and date values.

The following three special purpose services are also provided for use as remote procedure calls.

PCS_Execute synchronously or asynchronously executes the specified host command as a separate process.

PCS_TimeOfDay returns current time and date information from the host machine.

PCS_GetHomeDir determines the home directory of a given user based on his host account. For a brief discussion of home directory, see "LOGIN" on page 22.

PCVENUS

PCVENUS is the PC-DOS analog to the Andrew VENUS component. Its function is to trap and monitor all PC-DOS file system calls and intercept those that reference network drives or files on network drives. The intercepted system calls are then translated into an appropriate sequence of remote procedure calls to PCSERVER functions. Forty-five separate PC-DOS operations are handled in this fashion using the twelve file system services provided by PCSERVER.

PCVENUS has no direct interactions with users; all of its functions are performed in response to PC-DOS file system calls from ordinary application programs or standard DOS commands. Once executed it makes itself an extension of PC-DOS using the "terminate and stay resident" mechanism. It contains about 9000 lines of IBM PC Assembler code.

LOGIN

It is the LOGIN program that creates SNAP conversations for use by PCVENUS. LOGIN may also be used to issue the SNAP_BeginConv call on behalf of distributed applications (such as CUI) as well.

When logging in, the user supplies the name of a service machine, the server program to be used (currently PCSERVER or MESSAGESERVER), Andrew account information, and optional conversation parameters governing the use of encryption and protocol time-out intervals. Many of these inputs can be defaulted.

The user's Andrew account designates him or her as the owner of a unique VICE directory referred to as a "home directory". Unless the user specifically indicates otherwise, files and directories subordinate to the home directory can be examined by others but only updated by the user. Associated with the user's home directory are

- VICE space quota limitations.
- A Mailbox directory. Arriving mail is deposited here, but only the user can read it.
- A public directory. Files and directories subordinate to this one can be both read or written by any user.
- A private directory. Completely invisible to all but the user.
- An Andrew "preferences" file, containing among other things, the name of a default system printer for hard copy output.

When a user logs in to the PCSERVER, LOGIN selects the next available drive letter and associates it with the user's home directory to create a network drive. The user may override this process entirely by selecting a different drive designator or specifying any VICE directory for which his or her Andrew account will permit access.[8]

[8] Most users have read access to most VICE directories.

LOGIN will create up to five network drives and establish a unique SNAP conversation for each one. Typically, each conversation would provide access to different VICE subtrees, but this is not enforced in any way. Since each conversation requires a separate SNAP_BeginConv call, different parameters may be associated with each network drive. For instance, encryption could be specified for one network drive, but not for another. Or, different Andrew userids may be used to obtain different access rights to the same VICE directory, depending on which drive designator one chose to use when referencing a file.

LOGIN supports both an interactive (panel-driven) interface and a command line interface so that LOGIN commands may be included in DOS BAT files. A utility program (called LOGGEDIN) that will set the DOS errorlevel indicator to reflect whether or not a conversation with a given server already exists can also be used in a BAT file to bypass the LOGIN command when it is not needed.

CUI

The CUI (Common User Interface) is a command-oriented interface to the Andrew mail and bulletin board system. Most users with VIRTUE workstations use an alternative visual interface program designed to work with the Andrew Window Manager. The CUI was intended to provide equivalent function for users of non-VIRTUE workstations. Both interfaces to the Andrew mail and bulletin board system use the same server program (MS). MS is not aware of the distinct interface styles employed by its clients. Although the CUI has also been available to VIRTUE users for some time, it is a recent addition to the PC Server package.

For those functions that involve message composition the PC version of CUI permits the user to select his favorite PC editor, which is supplied with a small template file for the user to edit. In fact, using CUI, PC users may also use their editor to inspect or modify any accessible VICE file, regardless of text format or name length, without an explicit conversion step. This is because the CUI does all of its VICE file manipulations directly using the SNAP remote procedure call protocol, not the PC-DOS file system interface provided by PCVENUS. Prior to invoking the user's editor, the data is obtained, automatically converted to PC text format if necessary, and placed in a temporary PC-DOS file. This is the file the user's editor actually sees.

CUI is a good example of a user-level SNAP application program (PCVENUS, although a SNAP application, appears to the user to be an extension of PC-DOS, and has no externals of its own). It is expected that over time additional SNAP applications programs may appear, so that an increasing number of Andrew capabilities will ultimately be available to PC-DOS users.

CURRENT STATUS AND SUMMARY

The PC Server has been available for general use at C-MU since the beginning of November, 1986. Two PC diskettes and a user's manual may be purchased through the C-MU Computer Store at a nominal cost. The diskettes contain all of the PC components discussed in this paper with the exception of CUI. The CUI application will be offered as part of an upcoming release.

The PC Server supports three type of local area network adapters, IBM token ring, 3COM Etherlink (Ethernet) and Proteon ProNET. Extensions to the campus network for new users and departments are generally based on IBM's token ring. To date, over 500 PC users have obtained token ring adapters to connect their IBM PCs to the campus network. Several departments are using private PC Server host machines in addition to those that are publicly available.

The PC Server is an obvious migration step for users with IBM PC-style workstations. It presents them with a low-cost opportunity to join the growing Andrew community and enjoy the benefits of the campus network, without abandoning the familiar PC-DOS environment and the current generation of application software.

BIBLIOGRAPHY

1. Defense Advanced Research Projects Agency, Information Processing Techniques Office. *RFC 791: DARPA Internet Program Protocol Specification.* DARPA, September, 1981.

2. Leong. *The PC Server: User's Guide.* Carnegie-Mellon University, Information Technology Center, October 27, 1986.

3. Morris, Satyanarayanan, Conner, Howard, Rosenthal, and Smith. *Andrew: A Distributed Personal Computing Environment.* CACM, Volume 29 Number 3, March, 1986.

4. Satyanarayanan, Howard, Nichols, Sidebotham, Spector, and West. *The ITC Distributed File System: Principles and Design.* ACM Operating Systems Review, Volume 19 Number 5, December, 1985.

GLOSSARY

ACIS: Academic Information Systems. A division of IBM with responsibility for university marketing, development, and support.

Andrew: The software system developed at the ITC in support of the C-MU network, named for the university's founder, Andrew Carnegie.

APA: All Points Addressable. The term is usually applied to plotters or display devices designed for graphics applications.

BIOS: Basic Input/Output System. Computer firmware used to define formal, low-level interfaces to physical system resources. In the IBM PC product family the BIOS is stored in a read-only memory and also performs power-on self tests before booting the operating system.

BPB: BIOS Parameter Block. A PC-DOS control block that describes the nature, layout, and size of the FAT, root directory, and other internal file system elements.

C-MU: Carnegie-Mellon University.

DARPA: Defense Advanced Research Projects Agency.

DASD: Direct Access Storage Device. A generic term originally intended to apply to a variety of devices such as disks, drums, data cells, etc.

Datagram: The logical unit of information transmitted by the IP protocol.

Daemon: A program or subroutine that performs its function in the background, usually without requiring interaction with end users.

DoD: The United States Department of Defense.

FAT: File Allocation Table. The mechanism used by PC-DOS to manage the space available on a block device.

FTP: File Transfer Program. A program that provides the capability to transfer a file from one file system to another in real time using TCP/IP protocols.

IP: Internet Protocol. A low level DoD standard protocol that provides a basic, unsequenced, connectionless, end-to-end packet switching capability.

ITC: Information Technology Center. The organization put in place at C-MU to fulfill the conditions of the 1982 IBM/C-MU joint study agreement.

LAN: Local Area Network. The term applied to a network of discrete computers whose internal buses are joined by means of a high-speed, high bandwidth connection.

MIT: Massachusetts Institute of Technology.

Network Byte Order: A standard byte ordering used to exchange binary data among machines of differing architectures. It "just happens" to coincide with IBM 360/370 byte ordering.

Network Drive: The term used with the PC Server software to describe a connection to a host file system that supports an injective mapping of the PC-DOS file system.

RVD: Remote Virtual Disk. An example of a PC-DOS file system imbedded within a file of a foreign host file system.

SNAP: Simple Network Access Protocol. A private IP-based protocol that provides a secure, reliable, sequenced, half duplex conversation between PC Server components.

TCP: A high level protocol that provides a reliable, sequenced, connection between two programs over an IP network.

UNIX: An operating system originally developed by AT&T and distributed in source form to universities for a nominal charge.

VICE: The distributed file system component of Andrew.

VIRTUE: The user-interface component of Andrew that requires the use of an advanced function workstation with a mouse and APA display.