A Distributed File-Server for a Personal Computer Network

D.D. Cowan
Department of Computer Science

F.D. Boswell
T.R. Grove
Computer Systems Group
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Research Report CS-84-54
December 1984

A Distributed File-Server for a Personal Computer Network

by F.D. Boswell, D.D. Cowan and T.R. Grove

Computer Science Department and Computer Systems Group

University of Waterloo

1 Introduction

The Computer Systems Group at the University of Waterloo has developed a computing system called Waterloo microNet which is based on a network of microcomputer workstations. This network is used to support educational and research computing and as a test-bed for further investigation of networks of personal workstations.

The objective of this network is to make powerful computing systems economically accessible to students and faculty, while using commercially available computers and communication mechanisms. Since there is a constant demand for more and more computing resources within universities it was attempted to design the microNet system to offer inexpensive and easily maintainable computing.

Personal workstations are valuable computing resources and can be used for many computing tasks. Since the computing power and the user interface are in close proximity, definite advantages follow:

- ability to provide powerful, friendly user interfaces: for example, the coupling of inexpensive colour graphics to a powerful inexpensive CPU in a workstation allows development of interfaces tailored to the user and the environment.

- load independence: the real CPU time taken to complete a given computing task is related to the length of the task, not the load currently on the system.

The network connection extends the power of the workstation by allowing:

- data sharing: file-servers in the network can provide the ability to share information in a controlled manner and provide services such as electronic mail.

- resource sharing: scarce or expensive resources such as high-quality printers or photo-typesetters, or devices such as large-scale, cost-effective disks can be made accessible to a large community of users. Even scarce human resources such as operators and system programmers can be shared.

The microNet network consists of a distributed file-server which supports a tree-structured file system and a large number of personal workstations. The file-server is used to store the system and applications software and user files of all types including character and binary files. Files are either accessed in stream mode (continuous access of small pieces) or downloaded (one access to the entire file) from the file-server. The file-server operates on files as opposed to "virtual disks": in this sense is a true file-server rather than a disk-server.

The file-server was designed to support a large user community consisting of both students and faculty. From a user's perspective the file-server is similar to the UNIX[Ritc78,Bour78] file system, but there are fundamental underlying differences. The file-server can be distributed over several different computers in the network, thus providing flexibility. For example, file access to both systems and application software and user files can be optimized. Also, each user is assigned a file-space limit, thus forcing the user to make efficient use of file-space. This file-space limit is enforced continuously during a session not just at the time a user logs on or logs off. Finally, the file-server has several added features to provide a robust environment. We have found that these last two design features are particularly important when this service is being provided to undergraduate students.

The original version of the network has been in operation since January 1981 and has undergone constant refinement. Current versions of Waterloo microNet in use at the University of Waterloo support approximately 3000 separate users and 200 workstations.

The file-server runs as part of an operating system called Watsys which was also developed at the University of Waterloo by the Computer Systems Group. Watsys is a message-oriented operating system and has an internal structure similar to the one used in Thoth[Cher79,Cher80] and Port[Malc83]. Although Watsys is primarily supporting a file-server function, it is a general-purpose real-time operating system with language processors, editors and other facilities.

The file-server and the Watsys operating system were designed with three important properties in mind, namely efficiency, distributability and portability. Since this was a production system with a large user population and potentially heavy file access it was necessary to create a system which would access files in an efficient manner. Also, the volatile nature of computer technology and hence the constant availability of better CPU, network and communications hardware forced us to try to create a file-server system with two other design goals:

- the ability to locate files at appropriate nodes in the network depending on the bandwidth available (so-called "distributability"), and
- the ability to move the system from one network to another with a minimal amount of rewriting. In fact our goal was to design the file-server and the operating system so that the cost of moving it was less than 10% of the cost of rewriting it.

Obviously these last two goals are closely related.

In order to make the file-server portable it was necessary to identify hardware dependencies. There are three such areas: the CPU, the peripherals and the network. The CPU difficulty was removed by writing the system in a systems programming language called the Waterloo Systems Language[Bos82] (often shortened to the acronym WSL; pronounced "whistle"). The WSL system is operational on 15 different computer and operating systems, including the IBM 370, DEC VAX, DEC PDP-11, IBM Series/1 and several microcomputers based on the 68000, 8086, 6809 and 6502 microprocessors. The lack of portability between peripherals and networks was handled by designing appropriate interfaces between the device drivers and the network driver so that the file-server saw a constant interface independent of devices and network architecture. The ISO Open Systems Interconnection Reference Model[Tann81] was used as a guide in the design of these interfaces. Watsys and the file-server consist of 50,000 lines of WSL code and moving the system to a new network architecture requires the re-implementation of about 2,000 lines of code.

Watsys and the file-server currently operate on three different CPU's ranging from a large mainframe to a microcomputer and on three different network configurations.

2 A Functional View of the File-Server

2.1 Introduction

This section presents a functional view of the file-server and describes the structure, file attributes, permission mechanisms and capacity constraints.

The Waterloo microNet file-server system can be viewed as consisting of two components:

(1) the file server and system software, and

(2) the network controller software.

2.2 The General Structure of the File-Server

The file-server stores and manages files on disk. It is a general purpose file system and provides the following facilities:

(1) "Tree structure" (or "hierarchical organization"). This allows files to be grouped together in directories for simple and clear organization of data. Directories may contain other directories to any desired depth.

(2) Each file has a specific user associated with it called the "owner". A file also has permission attributes which determine whether or not the file is accessible to other users or groups of users.

(3) Each user can be assigned an individual file space limit which is continuously monitored and acted upon during the time a user is signed on.

(4) A simple library facility exists which allows read-only directories to contain files which are accessible to all users of the system. Only designated library administrators can modify the library.

(5) A queueing mechanism exists which allows an entire file to be sent to a program. Print spooling and electronic mail are implemented in this way. In fact, a queue is just a special type of directory.

The user communicates with the file system through the network controller: a program which controls access to the system and handles user protocol validity. Parameters to the network controller are stored in files and the system may be tailored by editing the parameter files.

2.3 Files and Directories

A file follows the UNIX[Ritc78] model and is just a sequence or stream of bytes.

Simple file-names can consist of up to 20 characters. Numbers and alphabetic characters are legal and so are most special characters such as dot ".", hyphen "-" and dollar sign "$". Blanks are not allowed and the slash "/" has a special meaning. File names are case-sensitive.

There is a special type of file called a directory which is a collection of files. Directories may contain directories so there can be several levels of hierarchical or tree-structured organization. The directory at the top of the tree is called the root of the file system and is represented by a special name consisting of only a slash "/".

Any directory in the tree can be specified by a filename which indicates the path from the root to that directory.

One directory may be designated as the current directory. A filename beginning with a slash is relative to the root and is called an absolute filename. A filename not beginning with a slash is relative to the current directory and is called a relative filename.

2.4 File Attributes

Files have the following attribute information associated with them:

(1) filename

(2) date of last modification

(3) owner identification number

(4) permissions

(5) file descriptor number

(6) file type (data file or directory) and resources (file size for data files and file space limit for directories)

Each person who may store files on the system is assigned a unique identification number called a "userid". Userid 0 is special and has complete permissions on all files. The userid of the owner of a file is called the owner identification number of the file.

File permissions determine who, other than the owner, can read and write a file. In general, permissions on a file can only be modified by the owner, although userid 0 (the "super-user") can read, write or modify the permissions of any file.

Each directory can be assigned a limit on the number of 1024 byte blocks available to files stored in that directory.

Devices are just another type of file. They are referenced by special names beginning with /dev. Even though the devices are conceptually under the root directory they will not be displayed when the files and directories under the root are examined.

3 Watsys

The description in the preceding section presented the user's view of Watsys and the microNet file-server. There are many topics for discussion which arise from the design and implementation of a portable operating system. In the descriptions that follow, the emphasis will be on the aspects of design which pertain to the implementation of the microNet file-server.

3.1 Watsys Organization

3.1.1 Nomenclature

Before the Watsys operating system is described, a definition of some of the terminology is presented.

process Watsys considers a process to be a single "thread" of execution, that is, a single sequence of instructions and associated data.

message A message is an object used for communication between processes.

3.1.2 Process Structuring

Stated succinctly, Watsys is a message-oriented system. The various operating system services are implemented as individual processes, and these processes communicate with each other by transmitting messages. If a process transmits a message to another process, it is normally blocked (it temporarily ceases execution) until the process to which the message was transmitted responds to the message. It is in this fashion that inter-process synchronization is achieved. Processes are dynamic entities, that is, they may be created and destroyed as required. The message-passing scheme is described in more detail in a subsequent section.

Processes and inter-process communication are implemented in a manner similar to that described by Cheriton[Cher80]. Watsys is designed to provide inexpensive inter-process communication and process management (such as process creation and destruction), and this process-structured mode of programming has been used liberally throughout Watsys. For example, each peripheral device connected to a machine running Watsys will require at least one process, and each application (such as microNet) running on the machine may create many processes. Process management and inter-process communication is the responsibility of the Watsys kernel, which is described in the next section.

3.1.3 The Watsys Kernel

The kernel is a small collection of routines which support the Watsys process abstraction. The services provided by the kernel include:

- process creation and destruction;
- message-passing primitives;
- interrupt management; and
- real-machine memory management.

These operations are considered to be operating-system primitives; that is, they are not part of the process abstraction, but rather are considered to be equivalent to a hardware operation. The kernel itself is not able to use process abstraction, since it is the implementor of the abstraction.

The kernel also uses a lower-level abstraction, the message switch, which carries out the actual delivery of messages.

An important aspect of the separation of the kernel from the remainder of the system is that these critical kernel operations and the message-switch may be easily identified and optimized for performance, thus reducing any impact from the overhead caused by the process abstraction.

3.1.4 Message Passing

Messages in Watsys serve two purposes: inter-process communication and data transmission, and process synchronization.

The fundamental message-passing primitives are <u>send</u>, <u>receive</u>, <u>reply</u>, <u>forward</u>, and <u>send kernel</u>.

Process synchronization is achieved in the following manner: since these primitives are blocking primitives, a process which sends a message to another process is suspended from execution until the recipient replies to the message. Similarly a process can block on receive while waiting for a message to be sent. Thus, critical sections such as device contention and file access contention are resolved in a straightforward fashion.

Interrupts and exceptions are converted by the kernel into messages so that they may be processed within the process abstraction model.

The message-passing primitives are designed to facilitate <u>thin-wire communication</u> between processes. Since no shared data space is involved in message transmission and reception, the processes can operate in different computers. This distribution capability is discussed in the next section.

3.2 Distributing the File-Server Over Multiple Machines

The portability of Watsys -- the attribute that the system can run on several different machines -- is important for two reasons:

- It provides a uniform base upon which applications such as microNet can be built.
- It facilitates the implementation of a <u>distributed</u> system. In this sense,

"distributed" means that several machines (not necessarily all of the same type), each running Watsys, are viewed as one large Watsys system.

A distributed Watsys system consists of a number of individual kernels (one per machine) which maintain a peer relationship amongst themselves. A kernel may create a process on its machine only, and processes may not migrate between machines. At this stage of development, applications such as microNet must be aware of the topology of the distribution, and inter-machine communication must be done at the file-system level, rather than the process level.

Since the file-server can be distributed over a number of nodes in the network, files can be placed to optimize network performance. For example, user files which are dynamic in nature can be centralized in one location for ease of access, while large software packages which are seldom updated can be duplicated at several nodes of the network. This means that downloading of software packages can occur over local high-speed transmission links while smaller volume user files can be accessed over longer and usually slower communication links. In other words, we place the files in order to make efficient use of current bandwidth. Of course as faster transmission media become available, file placement strategies can evolve to meet the new demands.

3.3 The Watsys File System

In a discussion of file systems, two issues which are of particular interest in a file-server application are data integrity and security.

It is inevitable that computer systems are subject to various failures and interruptions, such as hardware failure, operator error and software failure. It is desirable that such failures have only a minor impact on the integrity of the data in the system, that is, that the loss of data is kept to a mininum. The Watsys file-server achieves this goal by keeping two copies of a file during the updating sequence (the existing version of a file is not deleted until the new version of the file has been successfully written to backing-storage). If a system interruption occurs at a critical point in time, data loss in a file is restricted to the changes which have been made since the file was last updated.

The security, or permission facility in the Watsys file-server is implemented in a fashion similar to many tree-structured (hierarchical) file systems. When a file is created, it has associated with it an owner identifier. The owner identifier may represent a single user or a group of users. The owner of the file may always read and write the file, and further, may grant read and write permissions on the file to a pre-established group of users (group permission), or to all users on the system (global permission). If global permission has been granted on a file, then any group permissions are irrelevant.

Above and beyond these permissions, Watsys also has the concept of so-called "super-users". A super-user may override any permissions of its domain: the global super-user may modify any permissions in the entire system, and group super-users may modify permissions in their group.

3.4 File System Performance

Since most of the activity in a file-server application is concerned with accessing files, the performance of the file-server with respect to file-access times is of interest. Watsys utilizes two major techniques designed to improve the speed of file accesses.

3.4.1 File System Caching

File system caching is a technique whereby recently-accessed blocks of backing-storage are kept in memory, so that backing-storage access (and the inevitable associated delays) may be minimized.

Given that each block of backing-storage is uniquely identified, the Watsys file system caching method may be described as follows:

(1) If a block of backing-storage is to be read, the file system first searches the cache to see if the required block is already present in memory. If it is, then no access to backing-storage is required. If the block is not found in the cache, then it is read in from the backing-storage device, placed in the cache, and then given to the requestor.

(2) If the block is to be written, it is placed in the cache, and then immediately written to the backing-storage device.

(3) The cache is maintained using standard LRU techniques.

An interesting effect of this scheme which should be noted is that since blocks which are being written to backing-storage are always actually written immediately (with a copy left in the cache), many problems relating to the integrity of the file system in the case of system failure are avoided. For example, a user can not sign off the computer system without knowing that the file-space limit has been exceeded or that some other storage error has occurred. This method contrasts with "write-behind" schemes where file system errors may not be detected until after the user has completely left the system. This technique of writing immediately seems to decrease the total efficiency of file operations by about 10%, but we felt this was an acceptable tradeoff in our applications.

3.4.2 Directory Structure

In hierarchical file systems many accesses to backing-storage may be required in order to read a file. This is undesirable because the inherent physical properties of backing-storage technology make backing-storage accesses relatively slow. A design criterion of Watsys was to minimize the backing-storage access required to read a file. In particular, it was decided to implement the concept of a "current directory", and to optimize the access time to a file in the current directory.

The hierarchical structure of the file system (also referred to as the "directory structure") is stored in files on backing-storage. A "directory file" at some arbitrary point in the hierarchy contains information pertaining to the files stored at that point in the hierarchy.

In normal circumstances, an application program maintains a "current directory" which is used as the "starting point" for filename searches. For example, the interactive command processor, or "shell", establishes the current directory for each user. Also, the microNet network controller maintains a current directory for each communications line.

The information stored in a directory file allows it to be used as an index to the actual location of a data file. In the best case (accessing a file which is in the current directory), three accesses to backing-storage are required in order to access a file:

(1) reading a block of data from the current directory file;

(2) reading the file descriptor for the file (a list of data block numbers);

(3) reading the first block of data from the file.

The effect of the file system caching scheme on the performance of this file access method is substantial, since the data blocks for the current directory files tend to remain in the cache.

4 Summary

4.1 Current microNet Implementations

As mentioned in the introduction, the microNet file-server described in this paper is in use at the University of Waterloo. The current implementation consists of an IBM 370 (a 4341-2 mainframe running Watsys in a virtual machine under VM/SP), six IBM Series/1 mini-computers (each running Watsys) plus approximately 200 microcomputer workstations (IBM PCs, Commodore SuperPETs and others). The 370 mainframe is used as the primary file-server (and backing-storage), and the Series/1 mini-computers are used as smart communications front-ends (30 to 40 workstations each). This arrangement permits users to migrate between the various types of workstations and to access their files without regard to which particular workstation is being used. Preliminary performance measurements on the total system indicate that based on the current computing load approximately 30% of the resources of the mainframe are being used to support microNet functions. Prior to this configuration there were three independent Series/1 networks each running a file-server. When the system was integrated with the mainframe, the users were not even aware that the system had been upgraded, attesting to our ability to maintain a constant environment.

File-servers are implemented on both the mainframe and mini-computers and files are placed to obtain optimum file-server performance. Because of the portability and design of Watsys and the file-server the messages sent between two physically distinct file-servers are identical in format to messages which are routed inside a single file-server.

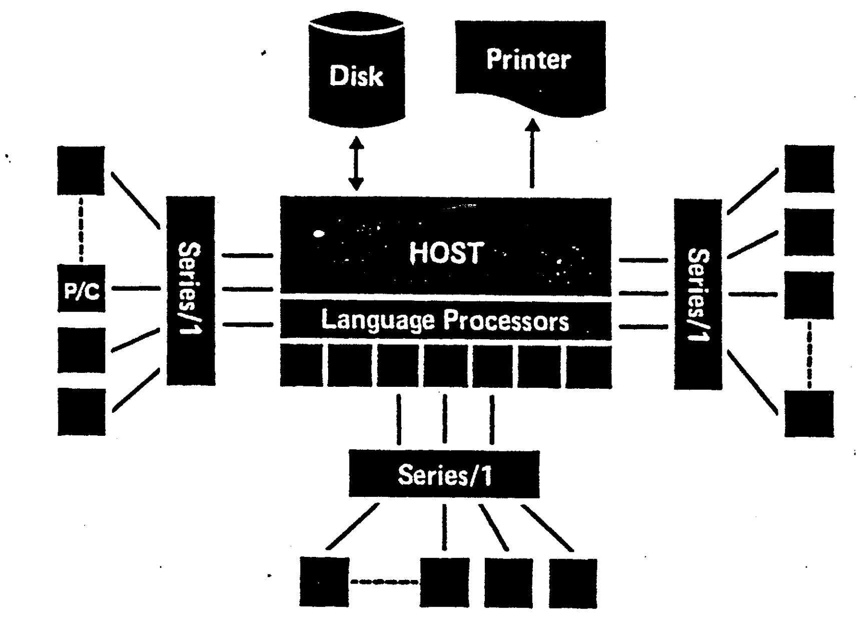

Figure 1

A configuration for the current system is shown in Figure 1. The physical communication paths in this system are:

- 9600 baud RS-232 serial communication lines between the workstations and the Series/1 mini-computers. Since these are relatively low-speed lines, they may extend over large geographic distances. The use of RS-232 lines is a design compromise: they are relatively cheap, simple, commonly available, and compatible with current telephony equipment, but they are not very fast when there is a requirement for large-scale file transfers such as downloading a software system. Our design of microNet is independent of this constraint and we are constantly seeking better communications mechanisms.

- High-speed (300K byte/sec) channel attach between the Series/1's and the 370 mainframe. These lines may only be used over a short distance (approximately 200 feet).

Another version of the microNet system has approximately 30 workstations and an IBM Series/1 mini-computer as the primary file-server.

4.2 Directions for Further Development

Development work currently in progress includes various Series/1 to Series/1 communications facilities, and low-speed (56 kilobit synchronous) long-distance communication facilities for the Series/1 to mainframe communication.

There are three major areas for future development. The first is to implement a distributed kernel for Watsys, rather than the peer-related multiple-kernel version that exists now. Such a development would permit the implementation of a microNet file server without a mainframe, since file sharing across a group of Series/1's will be possible.

Second, investigations are being made concerning the feasibility of replacing the Series/1-to-microcomputer communications with more modern (faster) technology. Since the microNet environment is portable with respect to networking facilities, this area of development tends to be a matter of examining new developments from the hardware manufacturers both from a technological and economic perspective, and then implementing the appropriate device drivers.

The third area for future development is the implementation of Watsys on the various microcomputer workstations. In particular, there are several high-speed communication peripherals already available for the IBM PC, and the implementation of Watsys on the PC would allow a microNet file-server to be implemented on a network of PC's without the need for either a mainframe or a mini-computer communications front-end. One typical configuration is shown in Figure 2. Files are stored on the personal computer file-server because the user accesses files in that geographical location most of the time. If the user temporarily moves and wishes to use the files then they can be accessed over the 56 kilobit line connecting the mainframe. Of course if the user relocates on a permanent basis then the files would be moved to a different machine. No software would be changed in the user's library and in most cases the user would not even be aware that the files had migrated to a new location.

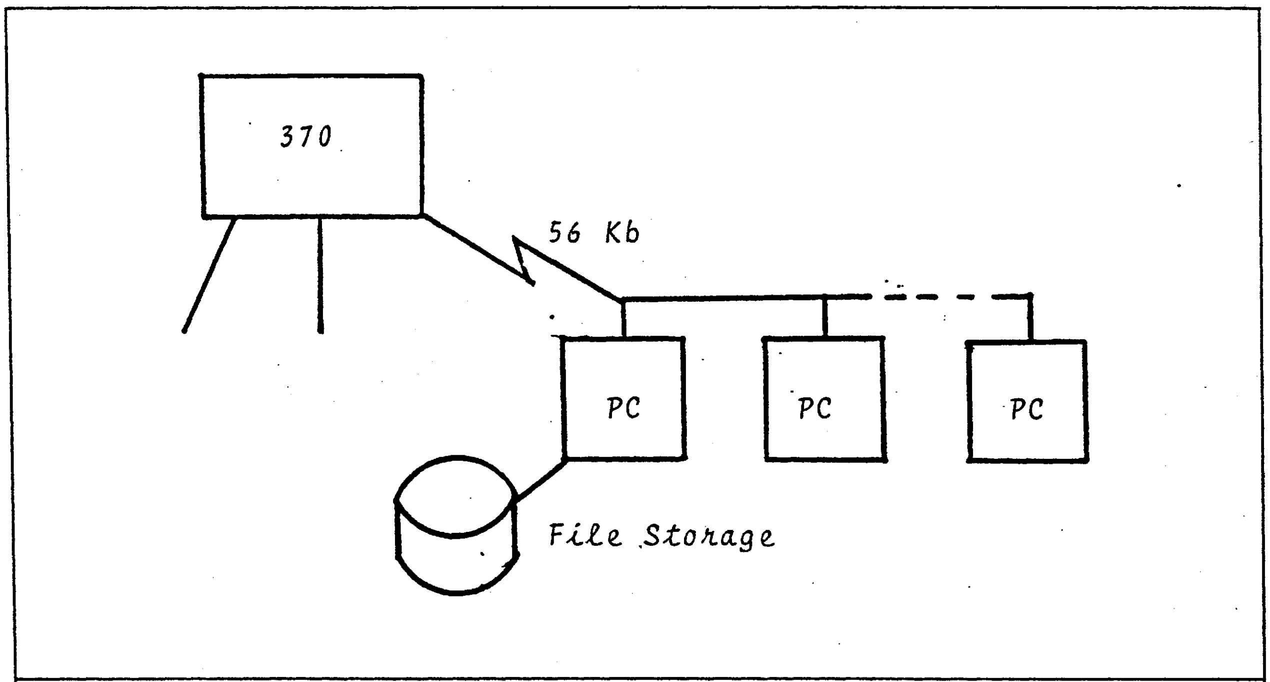

Figure 2

5 References

[Bos82] Boswell F. D.; Waterloo Systems Language Tutorial and Language Reference WATFAC Publications, 1982; ISBN 0-919884-00-8

[Bour78] Bourne S. R.; "The UNIX Shell" Bell System Technical Journal 57, 6, Part 2 (July-August 1978), 1971-1990

[Cher79] Cheriton D. R., Malcolm M. A., Melen L. S. and Sager G. R.; "Thoth, A

Portable Real-Time Operating System" Comm. A.C.M. **22**, 2 (Feb. 1979), 105-115

[Cher80] Cheriton D. R; Multi-Process Structuring and the Thoth Operating System Ph.D Thesis, Computer Science Department, Faculty of Mathematics, University of Waterloo; 1980

[John78] Johnson S. C. and Ritchie D. M.; "Portability of C Programs and the UNIX System" The Bell System Technical Journal **57**, 6, Part 2 (July-August 1978), 2021-2048

[Malc78] Malcolm M. A. Didur Phyllis A. and McWeeny Paul A.; Waterloo Port User's Guide Software Portability Group, Institute for Computer Research, University of Waterloo; 1983

[Ritc78] Ritchie D. M. and Thompson K.; "The UNIX Time-Sharing System" The BEll System Technical Journal **57**, 6, Part 2 (July-August 1978), 1905-1929

[Tann81] Tannenbaum A. S.; Computer Networks Prentice-Hall, 1981; ISBN 0-13-165183-8