CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (r.86/06)

Canada

A Microcomputer Network

for

Computer Science Education

by Peter J. Hickey

Thesis Presented to the School of Graduate Studies

in Partial Fulfillment of the Requirements for the

Degree of M. Sc. in Computer Science

University of Ottawa

Ottawa, Canada

ISBN 0-315-33331-6

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ACKNOWLEDGEMENTS

I am most grateful to the following characters, both real and imaginary for the various parts they played in helping me get this work done:

Jacques (hello sailor) Raymond - for everything.

R. Aftsman - for the inspiration.

Louis the Wizzard - for ideas, help, etc.

Steve Richards - for answering questions.

JP_1 and JP_2 - for pushing the buttons.

Spareparts Wilson - for the CPN Users Manual.

Digital Research - for the problems in CP/Net.

Pat & Roger - for burning 100 EPROMS.

In addition I would like to thank all the students that have been using and enjoying CPN.

Table of Contents

I What is CPN 1

1.1 Network Organization 2
1.2 Workstation Hardware 5
1.4 Workstation Software 6
1.5 A student's view 8

II Requirements of the System 13

2.1 Academic requirements 13

III Why a microcomputer network 17

3.1 Microcomputers Advantages 18
3.2 Microcomputer Problems 21
3.3 Microcomputer Network 23
3.4 Microcomputer Network Disadvantages 24
3.5 Conclusion 26

IV CP/Net: A Commercial Microcomputer Network 27

4.1 Introduction 28
4.2 Microcomputer Operating Systems 28
4.2.1 CP/M overview 29
4.2.2 BDOS Details 32
4.2.3 BIOS Details 33
4.2.4 The Boot Process 35
4.2.5 The Disk System 37
4.2.6 Putting it all together 37
4.3 CP/Net: The Microcomputer Network 39
4.3.1 CP/Net 40
4.3.2 CP/Net Implementation: Workstation . 41
4.3.3 CP/Net Implementation: Server 49
4.3.4 CP/Net Problems 51

V Design Philosophy 56

5.1 Network and Operating System Requirements . 56
5.1.1 Network requirements 58
5.2 The RAM DISK 59
5.2.1 RAM Disk Benefits 60
5.3 Network independent boots 61
5.4 Log on process 62
5.5 Fileserver Network 66
5.6 Virtual disks 68
5.6.1 Virtual Disk Implementation 70
5.6.2 Virtual Disk Advantages 71
5.7 Printing 72
5.8 Concluding Remarks 73

VI Software Details 74

6.1 Fileserver Requests 74
6.2 The fileserver 76
6.2.1 Logon 77
6.2.2 Open requests 78
6.2.3 Load System 79
6.2.4 Read and Write routines 80
6.2.5 Spool 81
6.2.6 Other Server Request 82
6.3 Fileserver Data Structures 84
6.4 Workstation Software 86
6.4.1 CPN BIOS 86
6.4.2 Disk Operations 87
6.4.3 Spooling 88

VII Network Details 90

7.1 A Polled Network 90
7.1.1 Message Interchange 90
7.1.2 Packet Format 95
7.1.3 Analysis of the Polling 96
7.2 Inter-fileserver Communication 97

VIII Concluding remarks 99

8.1 Problems with CPN 101
8.2 Future work 101

Appendix

A. State transitions for message interchange 104

B. Message Formats 106

References 108

List of figures

1.1 Physical interconnection 2
4.1 CP/M memory organization 30
4.2 CP/Net memory organization 42
4.3 Control flow for serial requests 46
4.4 Control flow for disk requests 48
5.1 Logical flow of control for disk request 70
6.1 Account and user record format 77
6.2 Pointer block 84
6.3 System structures 85
7.1 Message interchange 92

List of tables

1.1 Network sizes and coulours 1
1.2 Approximate load times 3
5.1 128 byte vs 2K byte transfer times 65

Chapter I

What is CPN

CPN is a microcomputer network servicing the first and second year students in the Computer Science Department at the University of Ottawa. As opposed to being an "off the shelf" product, designed to satisfy every type of user's needs, CPN is designed specifically to meet the requirements of its users. These requirements are based on the first and second year courses as specified by the department's curriculum committee.

The network consists of diskless workstations linked together in a network to allow the sharing of printers and Winchester (hard) disks. Students have their private area on the Winchester for storing their files, and are not bothered with floppy disks.

The workstations are running the CP/M operating system. CP/M is a simple yet popular operating system, and this provides a base on which to run a wide range of application programs.

1.1 Network Organization. There are several ways of viewing the network. In some respects, it is a single network: every workstation is linked to a fileserver, and the fileservers are all linked together. To the students, it appears to be two networks; a first year network and a second year network. Even though all workstations are physically linked together, the software does not allow first year students to use the second year workstations, and visa versa. Finally, it may appear to be five networks. There are five fileservers and a string of workstations is connected to each. For our discussions, we will consider "a network" to be a single fileserver and its printer and workstations.

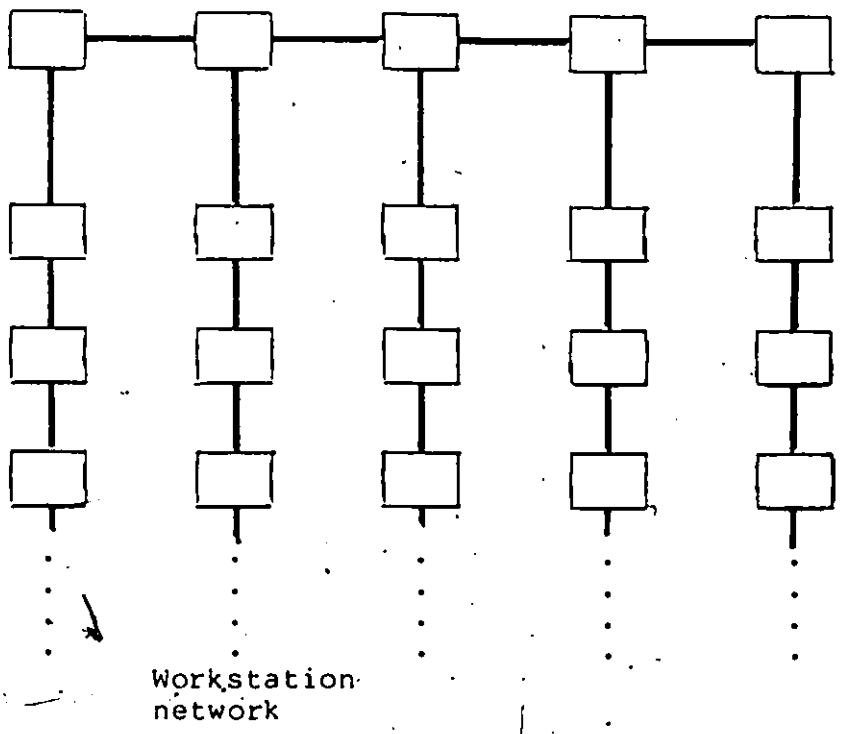


fig. 1.1
Physical interconnection

Lets now look at the five networks. Each of these networks is identified by a colour. Two of the networks are for second year students, and the other three are for the first year students. Each first year network has 14 workstations on it, and each one for the second year has 19. There is no magical reason for this choice of numbers, it initially came from the physical organization of the first year room, then expanding the lab to second year, and finally finding out that more workstations were needed for second year than had been expected. What is really important is that we want to have 42 workstations for the first year students, and 38 for the second year students. These numbers come from the number of students and the expected amount of work per student. The networks are summarized in table 1.1

NETWORK	Year	Number of workstations
BLUE	1	14
ORANGE	1	14
RED	1	14
VIOLET	2	19
GRAY	2	19

Table 1.1
Networks Size and Colours

Each of these networks has one fileserver and one printer. The fileserver contains a Winchester disk which stores the application software and is the permanent storage area for the students' private files. Each student is given a certain amount of space for their files. The amount of space is determined by dividing the total amount of space available by the number of students. We basically let them share all the space. It comes to about 100K bytes per student.

Initially, these five networks were independent. A student having an account on one network, and would have to find a workstation on that network. This caused several problems. At times, one network may have been relatively empty, yet a student would be unable to work because his/her account is on a network that was full. More serious problems would occur with second year students taking several courses. There are scheduled lab times when students are guaranteed to have a workstation. There is a good chance that in each scheduled lab several students couldn't get their workstation.

Because of these problems, the fileservers were linked together. This link is a seperate wire. Data transmitted from a workstation to a fileserver does not travel on this wire. Connecting the fileservers made the five networks into a single network. A student could log on at any workstation and access his/her files.

This linking together of the fileservers gives us another way of looking at the network(s). In talking about the network, it often makes things more simple if we consider two types of networks: the workstation network, and the fileserver network. We have five workstation networks, and one fileserver network. The two types are physically seperate, and the information traveling on them is different as well, so we are justified in thinking of them as they are seperate networks.

On the workstation network, each workstation can only communicate with the fileserver. This restriction is imposed by the software, and not the hardware. Although it is physically a bus

organization, logically it appears to be a point to point scheme organized in a star, with each workstation communicates only with its fileserver.

The fileserver network handles only requests from one fileserver to another fileserver. User requests are not passed directly to another fileserver, but are first processed by the local fileserver, and the request is only sent to another fileserver when absolutely necessary.

The only time the fileserver network is used is for accessing a student's files. System software and printing services are performed on the local fileserver. As was mentioned above, students for a given year should use their own equipment. First year users are prevented from logging on in the second year room and visa versa. Only privileged users have total network access.

1.2 Workstation Hardware. The workstations are DY/4 Challengers, the fileservers are a DY/4 Orion VIs, and the printers are Datasouth 180s. The computers are STD-bus based machines.

Each challenger contains 3 boards: memory, CPU, and a communication card. The CPU board contains a Z80 CPU, SIO, and CTC. The SIO has 2 channels which are programmed for asynchronous serial communication. Channel A is used for reading from the keyboard and writing to the screen while channel B has RS-232 drivers and is attached to a 25 pin connector at the back of the machine. The CTC and channel B of the SIO are unused by the operating system and are often used by students for assignments

on interrupt handling and machine communication. The memory card contains 256K bytes of banked memory. The communication card has a Zilog SIO and DMA. The SIO is connected to the line using a modified RS-422 protocol to drive the line. Standard RS-422 has a transmit pair and a receive pair and, therefore, cannot handle multi-point communication. On our cards, the transmit and receive pairs are wired together as are the transmit and receive clocks. The communication is, therefore, half-duplex over two twisted pairs. Because of the DMA, messages can travel at 800K baud. They are sent using the HDLC mode of the SIO chip.

The fileservers contain similar CPU, memory, and communication cards. Channel B of its SIO is connected to the printer. In addition, each file server has a 20M byte Winchester disk, and an 8 inch floppy. The floppy is used for back ups and obtaining distribution software. It is not accessible to the students.

1.3 Workstation Software. CPN does not stipulate what software should run on the workstation. The workstations could be running any popular microcomputer operating system, such as CP/M, MS-DOS, UCSD, etc. CPN gives the user a virtual disk on the fileserver. The user may format this virtual disk any way (s)he likes. The fileserver need not have the same structured file system as the workstation.

Although CPN imposes no restriction on the type of software running on the workstations, they currently are all running Digital Research's CP/M operating system.

CP/M is not a particularly well organized, bug free, or user friendly operating system, but it is simple and has so far proved sufficient for our needs. CP/M

- provides a base on which runs our required application software,
- is simple,
- is easy to improve by adding some additional programs.

In addition, at the time of setting up the lab, the 16 bit microcomputer operating systems were not too well established. CP/M was more established, and is still the most popular operating system for 8 bit machines. There is a wide choice of application software available which runs under CP/M, and we have had no problem obtaining compilers for Pascal, COBOL, PL/1, etc. as well as LISP and Prolog interpreters. There is also a variety of support programs, such as editors and work processors.

The application software that is running at the workstation is:

- Turbo Pascal,
- Digital Research's PL/1,
- Microsoft COBOL,
- Nevada Fortran,
- C-80,
- Micro-PROLOG,
- Stiff-upper LISP,
- WordStar (For word processing),
- WordMaster (For program editing).

In addition, there are other utilities such as spreadsheets, databases, etc. that may be put onto the system if needed for

special courses.

All editors (Turbo editor, WordStar, and WordMaster) use the same key strokes for the basic cursor movement. This is so that the students would not be confused by having to learn several editors depending on the language or system they use.

1.4 A student's view. What is most important is how the network appears and performs from the user's point of view. We will look at a typical student session.

Before even starting, the student has documentation available. The CPN User's Manual is available to the students. This is a low cost manual written expressly for the students. It contains all the information that they need, in a clear concise format. It is not written at too high or too low a level.

On entering the lab, the student sits down at a work station and pushes the reset button on the back of the machine. (S)he is immediately prompted

Boot from network (Y/N) ?

A network boot includes logging on and initial loading of the application software. Initially the student must boot from the network. After responding yes, the student is then prompted for a userID and password. Assuming that these are entered correctly, the following menu appears:

The following systems are available:

A) Assembler
B) Pascal
C) PROLOG
D) LISP
E) PL/1
F) COBOL
G) C
H) ADA
I) Weekly special
Z) CP/M only

Enter desired choice:

The student then chooses the language (s)he wants, and the loading of the necessary programs begins. The compiler, linker, editor, runtime systems are all loaded. The amount of time to load depends on the size of the system being loaded and the load on the network. Table 1.2 gives some approximate load times.

Language	size (K bytes)	time (no load)	time (load*)
Pascal	72	7 sec.	38 sec.
LISP	52	6 sec.	30 sec.
COBOL	180	14 sec.	70 sec.

*Load means printer is printing and 9 stations are loading the same system.

Table 1.2

Approximate load times

After the application software is loaded, the workstation's operating system and a boot block for the RAM disk is loaded. The workstation then performs its boot and the student is sitting at a CP/M machine.

This CP/M machine gives the student three disks. The A: disk is the primary working disk. It is a RAM disk and is very fast.

The B: disk is the student's private disk. After working with the files on the A: disk, (s)he copies them to the B: disk for permanent storage. The C: disk contains some less often used application programs as well as files required for particular course. For example, a professor may tell the students to run their program using a special file for input. This file will be found on the C: disk.

In the course of their working, the student may get the machine into an endless loop, or an array may go out of bounds destroying part of the operating system. This will require the student to reboot the workstation. Depending on the amount of "damage" they did to their machine, there are several paths they may take to reboot.

The most basic reboot is the non-network reboot. There is a boot block and a copy of the operating system kept in memory. This is in banked memory and is not normally accessible to the processor. When the student pushes the reset button and answers "no" to the boot from network prompt, the copy of the operating system is copied into the processor's memory. This reboot takes a small fraction of a second, and will recover from most problems a student runs into.

If the student's program was swapping in and out banks of memory (this will only happen if they have an assignment on banked memory), or if a wild jump was performed and it happened to end up in some part of the operating system that manages the banked memory, there is the chance that (s)he could have destroyed the

boot block and/or the back-up copy of the operating system. In this case, the student can perform a "short logon". (S)he selects "CP/M only" from the menu of choices. This type of logon will take about 3 seconds, not counting the amount of time required for the student to enter the userID and password. This causes the copy of the operating system in the banked out memory to be replaced.

Now that we've looked at logging on and reboots, lets see what is done once the student has logged on and is working. The student then uses the COPY command to transfer a file from the B: disk to the working A: disk. (S)he may then work with this file, and no network accesses are required until it is necessary to either print the file, or to save it back on the B: disk.

Printing services are also designed for a student environment. The ideal listing for a student to hand in contains the source, compilation, and running of the program on a single listing. This shows the marker that the source listing is the same as the program that was run. The student issues the command SPOOL CON which causes everything written on the screen to be transferred to the printer as well. The student can then compile the program, type the listing, and then run it, producing a single listing to hand in.

After issuing the SPOOL CON command, the screen goes into inverse video so that the student can easily tell that printing is going on. Without this inverse video, a student may forget that the screen is being transferred to the printer, and send an entire

work session to the printer. The student also has a command to kill the spool file if (s)he accidentally runs a program with an endless loop in it while printing.

Finally, when the student is finished working, (s)he transfers his/her file from the A: disk to the B: disk for permanent storage, and issues the LOGOFF command. The LOGOFF command erases the student's files from the A: disk so that another student cannot come by and perform a short logon to steal them.

Chapter II

Requirements of the System

CPN was designed expressly to suit the needs of the first and second year students in the Computer Science Department at the University of Ottawa. Before looking at the design of CPN, we will examine these requirements so that we will be able to see how they influenced CPN. It should be again mentioned that these requirements come about from the courses given. These courses were decided on by the University's curriculum committee and do not necessarily represent the requirements at all universities.

2.1 Academic requirements. The computing needs of first and second year computer science students are different than the needs of other disciplines or real world users. Computer science students need a wide variety of capabilities which must also be very flexible.

The first year requirements are very simple. Courses are in problem solving and programming. There are not many demands on a system to satisfy their needs. A friendly well documented operating system, editor and a high level language, such as Pascal.

The compiler (or interpreter) should produce good diagnostics for compile time (syntax) errors, and there should be a good runtime system to aid the student debugging his/her program. Since the programs written are not for production, there is no need for an efficient or optimizing compiler, a fast one is preferable. In addition, there should be a word processor. Since word processors are a result of the "computer revolution", it is fitting that computer science students should be doing "written" assignments on them.

Second year requirements are more complex. They fall into the following categories:

- algorithms,
- programming languages,
- machine architecture,
- introduction to operating systems.

A machine that satisfies the first year students would also satisfy the needs for an algorithm course provided that the language has sufficient constructs, since more features (eg. dynamic storage, pointers, etc.) are needed.

The needs of programming language courses generate other problems. There must be a wide variety of programming languages available on the system such as PL/1, Ada, Prolog, LISP, etc. Flexibility comes into play here. As new and different languages are invented, they should be quickly available for the system. This implies that the operating system running on the machine should be a popular one.

Courses in machine architecture or design, if not taught completely theoretically, require a machine on which the students can get at the hardware. They should have access to devices on the system to experiment with, for example, to work with interrupts, or program a communication port, etc. The operating system must allow the students to get at the hardware. There is another requirement for the system which is implied when low level access is given to the students: Accidental or intentional mistakes at this level (or any level for that matter) should not cause it to crash affecting other users.

Introduction to operating systems courses, as well as having the same requirements as the architecture courses, require an operating system that is easy to communicate with. At this level, the student should be learning what services the operating system can provide, and how they can be used. The student should not have to be bothered learning complicated details and linkage protocols before being able to do any work pertaining to the course.

The system must be secure. In addition to the area mentioned before (students not being able to crash the system), there must be protection against copying files for the purpose of cheating. There must also be a way to limit resources. We should be able to prevent a student from accidentally or intentionally filling up the server's disk, or from printing files that are too large.

The University of Ottawa has an additional problem that most universities do not have. It is a bilingual university and, as

such, it would be desirable to have French versions of as much of the software as possible, as well as hardware (screens and printers) which support French characters. This in itself is not too much of a problem, but it must be an integrated system: The software and hardware must support the same character set.

Chapter III

Why a microcomputer network

The academic requirements will be satisfied by the software, but we also have certain expectations of the hardware. In this chapter, we will be looking at the advantages that a network of microcomputers has over minicomputers and to some extent mainframes. At this time, we are assuming that the necessary software exists, and we are looking at the microcomputer network pretty much from the hardware point of view.

We will remind the reader that our users do not need the great processing power of a minicomputer or mainframe for any single user, rather the processing power will be divided over many users running many small jobs.

We would like a system which:

- has a low initial cost,
- has low maintenance cost,
- has low operating cost,
- is reliable,
- performs well.

There are three types of systems that we can look at, mainframe, mini, microcomputers. Each one has good and bad points associated with them.

It is difficult to make a fair comparison against a mainframe because the mainframe is not usually owned by the department, but is a shared machine used by the entire university. It is often owned by the university's computing centre and is designed to satisfy the general needs of the university and not the specific needs of the CS department. Because of this, we cannot make a fair comparison of the price, although in areas other than cost, the mainframe has the same advantages and disadvantages as the mini.

3.1 Microcomputers Advantages. Lets look at what advantages microcomputers have over minicomputers or mainframes. Note that these advantages are from the point of view of our needs, and are not general advantages. These areas are:

- performance,
- reliability,
- all areas of cost,
- software maintenance is easy.

Of course, a mainframe or minicomputer has much more processing power than a microcomputer; at times when a single user needs this great power, the machine vastly outperforms the micro, but this type of use is not expected of our students.

As far as performance, microcomputers give constant response time no matter how busy the lab is. If it takes five seconds to compile a program when there is no one else in the lab, it will also take five seconds to compile the same program when the lab is full. This is because there is no sharing of resources; each user has own complete machine.

Microcomputers are reliable as well due to this distributed processing. There is no single point of failure that can bring down the entire "system". If a station breaks down, it effects only the user working at the station. A problem on a mini or mainframe, on the other hand, will prevent anyone from working.

A lab equipped with microcomputers can have equipment repaired much faster then a lab based on a single larger computer. There are several ways we can justify this. It is economically feasible to have several spare machines. When a machine dies, the lab technician simply replaces the entire machine, giving a down time of several minutes. It is usually not possible to keep spare parts for a mini on hand, and the department must depend on the response time of the manufacturer's service department.

Another way we can justify the fast repair time is that a department's technician will probably be able to repair the machines himself. A microcomputer is much less complex than a mini.

In the case of a minicomputer, a fair amount of expertise is required on the part of the system manager. A multi-user operating system is a complex thing. On the other hand, the single user operating system running on microcomputers is easily and

completely understood by someone with much less training. For example, our microcomputer network has its software maintenance done by second and third year students.

Finally, the purchase cost of the microcomputers is less than that of a mini. Consider the case of a VAX, which is a popular minicomputer used in many universities. A VAX/750 with 4 Megabytes of memory will support about 32 users at a cost of $150,000.00. This gives a cost of just under $5,000.00 per station. For a slightly larger system, we can look at a VAX/780. The University's computing center bought a VAX/780 for about $315,000.00 and is is supporting around 60 users. This price also comes to about $5,000.00 per workstation. A microcomputer can be purchased for less than $3,000.00.

The upgrade cost for a microcomputer lab is constant: the price of a workstation. The upgrade cost of a mini goes in steps. Consider the case of a VAX/750 supporting 24 users. Adding one more user requires the purchase of a DZ-11 multiplexor, which gives eight terminal ports. The cost for a DZ-11 is about $2,500.00. This the cost of adding the 25th user is the cost of the DZ-11 plus the cost of a terminal. The cost of adding the 26th through 32 user is simply the cost of the terminal, since the DZ-11 bought for the 25th user had 7 unused ports on it. Once we get to 32 users, we are pushing the limits of the machine, and instead of a VAX/750, we would need to go up to a VAX/780. This is very expensive.

Note from the last example, that it only cost a little over $300.00 per user, plus the cost of the terminal to add the last eight users. In the case of a minicomputer, we get the lowest cost per user when we push the machine to its performance limits. Indeed, we could put 64 users on a VAX/750 at a cost of less than $3,000.00 per user (Not including the cost of the terminal). The performance in this case would be totally unacceptable.

The fact that a microcomputer lab will let us upgrade at a constant cost, which is a nice feature for a growing computer science department.

3.2 Microcomputer Problems There are several problems associated with microcomputers. Some of these problems stem from the fact that they are distributed, while others problems are unique to microcomputers. These problems are:

- printing services,
- distribution of software,
- floppy drive maintenance,
- theft of software,
- keeping off unwanted software.

Since they are distributed, there is no centralized printer. There are two ways to get printing done. One is to equip every microcomputer with a printer. This is both costly and difficult to manage and maintain. The other way is to have several stations designated as printing stations; students work on the other stations, and when they need a printout, they go to the printing stations with their floppy and print their files. Problems with

this scheme occur when the lab is very busy. Rather than wait for an available workstation, some students will try to work at the printing stations thus preventing other students from getting their listings.

The stand alone microcomputers, like any machine, need software, but in their case it poses additional problems. The software is on floppy disks. We have several ways of managing this. We can leave floppies near the machine, and the student chooses the software that (s)he needs and uses the necessary floppy. In our case, we would need at least 8 floppies per second year machine, and one copy per first year machine. This means that we would have to maintain close to 400 floppies in the lab. Stolen or damaged copies would have to be replaced often.

Another option is to give each student a copy of the necessary software. This eliminates the need of maintaining the floppies in the lab, but it means a tremendous job of producing several thousand floppies.

A large lab with a great number of floppy drives would need a lot of maintenance; cleaning and adjusting drives, making sure students are careful when they use them, etc. In addition, we have to worry about the reliability of the floppy disks themselves. Students tend to purchase inexpensive floppies. Problems of lost assignments could be blamed on the machines when it is actually due to the low quality floppy. The reverse can also happen. What do you say to a student who has lost a weeks work due to a mis-aligned head on a floppy drive?

Theft of software is more of a problem with microcomputers than with minis: we would not expect students to try to steal a Pascal compiler from, say a VAX, but we have to worry about its theft from microcomputers. This is due to the fact that there is a chance that some students may have a similar machine at home, and may have a use for the software.

Finally, we have the problem of students bringing in their own software, more specifically, games. There is a wealth of games that run on microcomputers, and students could bring in floppies with games on it which could turn the lab into an arcade. Students wanting to work would be prevented due to others playing games. To make matters worse, a lab of this sort would attract non-students: people simply looking for a place to play games.

3.3 Microcomputer Network. The microcomputer network attempts to get the best of both microcomputer and minicomputer features. The workstations are almost entirely independent, but they linked together with a fileserver. The centralized fileserver serves as a software distributor. Lets look at how it solves our problems.

First of all, the problem of distributing software disappears. The student downloads the necessary software from the fileserver when it is needed. Note that for some uses, such as in an office environment, a microcomputer network would allow the sharing of data files. In our case, we do not want sharing of data, but sharing of resources: disk space and printing. The application programs are not shared, but distributed by the fileserver.

If we can have diskless workstations, we solve our other two problems. With no disks, there is no way for the students to steal software from the system, as well as no way for students to bring in their own games.

The diskless workstations provide some additional advantages as well. Diskless workstations with no mechanical parts to break down, and are more reliable. The ability to use a diskless workstation gives us a reliability advantage even over stand alone microcomputers.

If we use diskless workstations we have to provide an area on the fileserver on which students can save their files, since there is no way for them to transfer files to their own floppy. This means that we have to have a larger hard disk on the fileserver, but the cost of putting, say a 20 megabyte drive instead of a 10 megabyte drive is a small percentage of the cost of the fileserver.

3.4 Microcomputer Network Disadvantages. With a microcomputer network, we loose slightly in all areas that we gained with the stand alone microcomputers:

- cost
- Performance
- Reliability.

There is the additional cost of having a file server for every n workstations. In our case, this cost is less than $10,000.00 per fileserver. This gives us an additional cost of $715.00 for the

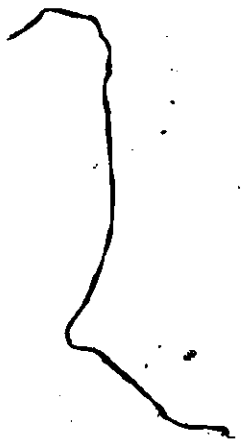
fifteen first year workstation, and about $525.00 for the nineteen second year workstations. We also have the cost of the network interface for each workstation. In our case, the network interface is about $450.00 per workstation. The total increase per workstation is now less than $1200.00. In our case, the cost per workstation was less than $4,000.00 which is still less than that of a minicomputer at $5,000.00 per workstation.

Although the system is still somewhat distributed, there is now a central point whose performance will decrease under load: the fileserver. Students are typically doing a short edit, compile, link and run while they are debugging their programs. This is fairly disk intensive and could slow down a file server. One way to greatly reduce the load is to transfer the compiler, editor, linker, etc. to the workstation once, thereby reducing the traffic on the network. If the workstations are diskless, we have to worry about a way to store these programs on the workstation. It may involve the addition of more memory at an additional expense.

Finally, with a microcomputer network, there are more problems with reliability than there are with stand alone microcomputers. If a fileserver dies, all the workstations connected to that server is out. We are still better off than in the case of the mini, since there are several fileservers per lab, and if one breaks down, the others will still be running. Although more expensive than the workstation, it is still possible to keep another fileserver around as a replacement in case of failure. Another possibility is to have a network set up in such a way so

that if one fileserver dies, the other could take over its load. Performance would suffer, but at least work could still be done.

3.5 Conclusion We now see, that for our type of usage, a microcomputer network offers us many advantages over stand alone microcomputers or minicomputers. Remember, though, that this is all based on the assumption that the software running will let us use the hardware in an efficient manner. We will soon see that CPN is software that offers this to us.



Chapter IV

CP/Net: A Commercial Microcomputer Network

At this point, we will take a little pause to look at Digital Research's microcomputer network, CP/Net. If the reader is familiar with microcomputer operating systems and microcomputer networks, (s)he may wish to skip this chapter.

There are several reasons for looking at this product. In the first place, it will be an introduction to the terminology which will be used later. Another reason for looking at CP/Net is that we will see how it does not and even <u>can</u> not solve some of the problems that we have. We will see that a different approach must be taken if we want to do things such as limiting resources, and providing security of files.

CP/Net is the only CP/M network software sold that is simply sold as a hardware independent network. Manufacturers of some machines sell <u>their</u> particular network but this is tied to their machine and will not run on any other machines. Corvus sells a network that runs on several machines. Although less hardware dependent than one running on a single machine, their Constella-

tion network is still hardware dependent.

4.1 Introduction. On larger computer systems, communication subsystems are usually done by having some tasks running in the background, and a certain amount of abstraction from the rest of the system can be achieved. On a single task microcomputer, any networking must be an integral part of the operating system. In order to understand the way that these networks are implemented we have to look in some details at the way microcomputer operating systems are designed.

We first examine, in some detail, CP/M, a typical microcomputer operating system. CP/M is the operating system after which most others are modeled, and therefore, is a good one to give the needed background. Once we see the design principles of microcomputer operating systems, we can then look at how the microcomputer network follows a similar design strategy.

4.2 Microcomputer Operating Systems. In the world of mainframe computers, the manufacturer of a machine also supplies the operating system: It is made for the computer. Microcomputer operating systems are usually written by a software producing company, and bought by the manufacturers of the microcomputers. Because of this separation, these operating systems are written to be hardware independent. This also gives the software company a wider market for their operating system.

In this section, we will look at a popular operating system for microcomputers to see how its design gives hardware independence.

Although we will only look at CP/M 2.2, other systems, such as MS-DOS, CP/M-86, and UCSD, follow the same style of design.

4.2.1 CP/M overview. CP/M currently is the most popular operating system for 8 bit microcomputers. It is produced by Digital Research (DR), and was the first microcomputer operating system developed to run on a variety of hardware configurations. This, in turn caused application software producers to develop their products for CP/M. The availability of this application software caused other manufacturers of machines to sell their machines with CP/M which in turn provided a larger market for software running on CP/M. Thus, today, there is vast amount of application programs which run under CP/M.

CP/M's hardware requirements are simple: an Intel 8080, Intel 8085, or Zilog Z80 processor, at least 24K bytes of contiguous memory, a console (terminal), and one or more disk drives. CP/M can also support a printer, an additional serial input device, called the reader and an additional output device called the punch. Although the operating system itself could run on a machine with as little as 8K bytes of memory, DR recommends having at least 24K bytes to do anything useful. In practice most application software requires at least 48K bytes.

CP/M version 1 only supports two 8" IBM format floppy disks. Later versions support up to 15 disks of any size or type: floppies and/or Winchester.

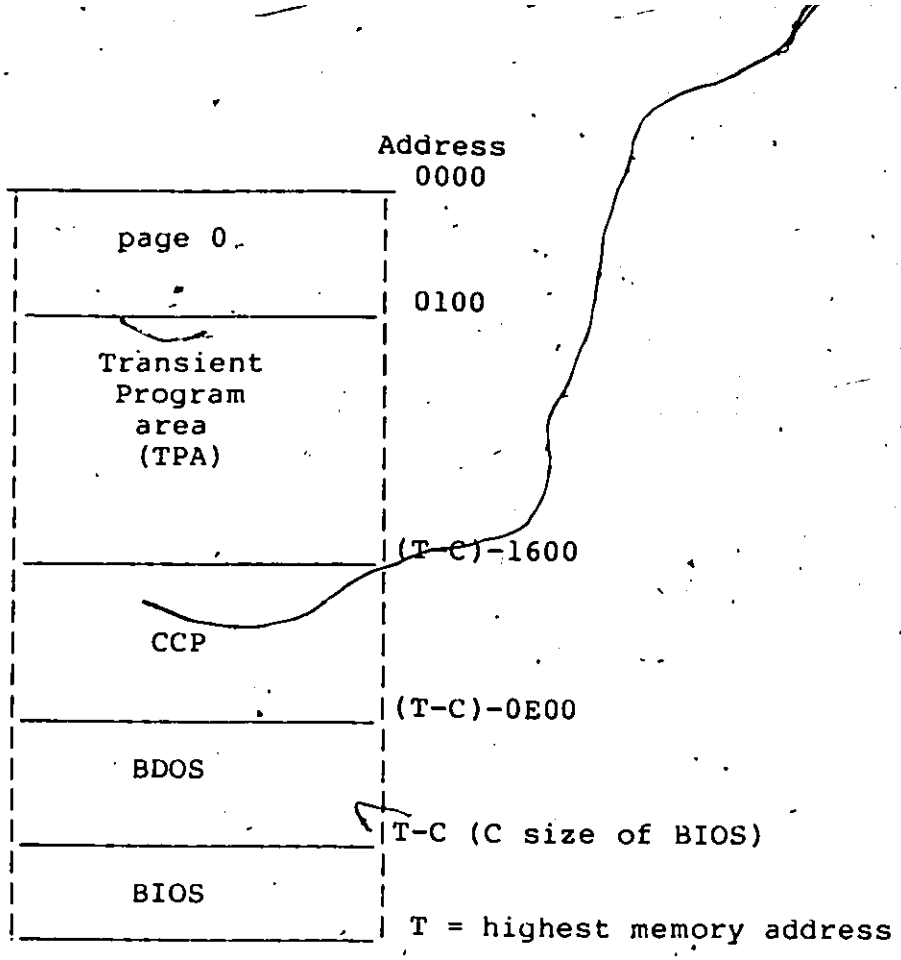


fig. 4.1
CP/M memory organization

The memory usage of CP/M is shown in figure 4.1. Page zero, as it is called, contains system constants, areas for passing parameters to application programs, and jumps to the BDOS and BIOS (These will be explained later). This was chosen to be at the low end of memory since all systems, no matter what their size, have these addresses and constant memory addresses can be used.

The next area is the Transient Program Area (TPA). This is where compilers, editors, etc. are loaded and run. This extends from address 100H up to the base of the CCP. The CCP, or Console Command Processor is the program which is the user interface to the system. It accepts commands from the keyboard and either

executes them, if they are resident routines, or loads the appropriate program from disk. Application programs are loaded starting at address 100h and are then called by the CCP. They can terminate by returning to the CCP with a return instruction, or by jumping to address 0, thereby causing a warm boot to be done. The warm boot process will be described shortly.

The BDOS, or Basic Disk Operating system is the interface between an application program and the hardware. It maintains the file system on disk, and provides I/O services to the application program. Calls to the BDOS are performed by issuing a call to address 5, which contains a branch to the BDOS entry point. This is why application programs can be transported in object form from one system to another: All I/O is done by a CALL 5 instruction.

So far, everything is hardware independent (assuming the proper CPU). Every system, no matter what the hardware configuration, runs the same CCP and BDOS*. The next section, the Basic Input/Output Section or BIOS, contains all the hardware dependent routines. It is the responsibility of the manufacturer of a machine to write a BIOS for his hardware configuration. Once this is done, the machine can run CP/M. The size of the BIOS is not fixed. It takes from the highest memory address down to as much as it needs. The addresses of the start of BDOS and CCP are determined from the start of the BIOS.

* Except for some systems such as DEC's Rainbow in which DEC has extensively modified CP/M to use their dual processors.

4.2.2 BDOS Details. As mentioned above, an application program should do all its I/O via calls to the BDOS. We can break the BDOS calls into 4 types: Console I/O, serial I/O, file I/O, and information. The information type of call is for obtaining some information about the system, such as the number of disk drives, or the version number of the operating system.

The BDOS gives the user a choice of several ways to do console I/O. These vary in the amount of formatting that is done, and can range from a call to test the status of the keyboard (ie Has a key been struck?) to reading a full line with limited editing features. The serial I/O calls do nothing but read or write a single byte to the serial device.

The bulk of the BDOS is in the disk system. The BDOS maintains the file structure on the disk. The application program has the typical calls.

- Open or close a file,
- Erase a file,
- Rename a file,
- Read/write sequential
- Read/write direct.

A file is a collection of 128 byte blocks. There is no concept of records, blocking, etc., supported by the operating system. It is the responsibility of the application program to do any of this itself. The size of the block comes from CP/M version 1.0 which used 8" floppy disks with 128 byte sectors. The disk may have physical blocks of any size, but this is transparent to the

application program and the BDOS as well. It is the job of the BIOS to make the disk look like it has 128 byte blocks.

The BDOS accepts a request to perform an operation on a file. This operation may involve several disk read or write operations. For example, a create file operation involves reading the directory to find the first empty slot for the directory entry. Then the new entry is rewritten. When a write request is issued, the BDOS takes care of finding a free block on the disk. The BDOS does the calculation of the disk addresses for the operations, but the actual I/O is done by calls to the BIOS.

4.2.3 BIOS Details. The BIOS is where all of the I/O is actually done. The BIOS starts with a jump table as is shown below. Since there is a well defined order to this table, the BDOS need only know the starting address of the table to find the address of any routine in the BIOS. Just as an application program is transportable from one machine to another because it follows the conventions for calling the BDOS, the BDOS can be transported from one machine to another because it follows its rules for calling the BIOS. This is the key to making CP/M portable over a range of hardware configurations.

```
BIOS+00     JP   COLD      ; Cold boot routine.
BIOS+03     JP   WARM      ; Warm boot routine.
BIOS+06     JP   CONSTAT   ; Console input status.
BIOS+09     JP   CONIN     ; Read byte from console.
BIOS+0C     JP   CONOUT    ; Write byte to console.
BIOS+0F     JP   LIST      ; Write byte to list device.
BIOS+12     JP   RDR       ; Read byte from READER.
BIOS+15     JP   PUN       ; Write byte to PUNCH.
BIOS+18     JP   HOME      ; Disk heads home.
BIOS+1B     JP   SELDSK    ; Select disk drive.
BIOS+1E     JP   SETTRK    ; Specify track to be read/written.
BIOS+21     JP   SETSEC    ; Specify sector to be read/written.
BIOS+24     JP   SETDMA    ; Specify disk read/write buffer.
BIOS+27     JP   READ      ; Disk read.
BIOS+2A     JP   WRITE     ; Disk write.
```

The COLD and WARM routines will be discussed in the next section on the bootstrapping process. The next five routines are for the serial I/O devices. These are straightforward and will not be discussed here. It suffices to say that all the BDOS serial I/O functions are done by calls to these routines, and that these routines read or write a byte to the physical device.

The disk routines are more interesting. In order for a disk operation to take place, the first call issued is to the select disk routine. This routine returns the address of a disk descriptor table or an error code if the disk does not exist. The disk descriptor table gives the size of the disk, number of tracks per sector in terms of 128 byte sectors, etc. It also

contains parameters chosen by the system designer such as specifying the size reserved for the boot blocks and directory, the allocation factor to be used, etc.

This is followed by calls to home, set track, set sector, and SETDMA. The first three calls specify the disk address that the data will be transferred to (from), while the last one specifies the memory address that the data comes from (goes to). Finally, there is a call to either READ or WRITE. It is only in these routines that a disk operation need actually be done. If the disk has physical blocks that are larger than 128 bytes, it is the responsibility of the BIOS to do the deblocking and make the disk appear to be a disk of 128 byte sectors. The BDOS expects disks with 128 byte sectors.

The important thing to notice here is that the BIOS is given a pair, (track,sector), which specify the unique address of the block to be read or written. It doesn't matter how it is physically implemented on the disk. Another point is that the BIOS also knows nothing of the file structure of the disk, it works at the level of raw blocks.

4.2.4 The Boot Process. The boot process is fairly straight forward in CP/M, but somewhat old fashioned; it is very inflexible. Other, newer microcomputer operating systems give a certain amount of flexibility and changeability to the system designer so that (s)he may easily modify the shell, or user interface to the operating system.

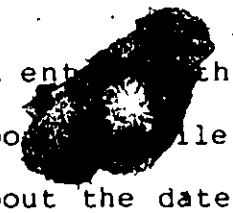
There is a basic assumption that the pushing of a reset button will cause a program in ROM to load the first sector of the disk into memory and pass control to it. This program must then load the BIOS, BDOS, and CCP. Control is then passed to the COLD routine of the BIOS. It is the responsibility of this routine to initialize I/O ports, tables, etc. The cold boot routine ends by branching to the CCP, which then initializes the disk system via BDOS calls. CP/M is then ready to accept user's commands from the console.

The CCP, BDOS, and BIOS are not files on the disk, rather they are at fixed locations (the first few tracks) on the disk. Since they are not files, it is not an easy job for the typical user to modify them, and their size is fixed, being limited by this reserved disk space. Under MS-DOS and CP/M86 these are files and a system designer could easily create a new large CCP that is very user friendly and tailored to particular needs.

CP/M has both a cold and a warm boot. The cold boot is the above mentioned initial bootstrapping process. The warm boot concept is very important. It reinitializes the disk system and is used for program termination. A jump instruction to the warm boot routine is at address zero of the memory. The warm boot routine reloads the CCP and BDOS into memory then passes control to the CCP. Remember, that an application program can (and often does) destroy the CCP to get more dynamic memory. It can also use the memory occupied by the BDOS as long as it doesn't need any BDOS routines. As long as the BIOS remains intact, an application

program can do whatever it wants, then jump to the warm boot routine to have the operating system restored. This is, in fact, how most programs terminate: with a jump to location zero.

4.2.5 The Disk System. CP/M allows for up to 15 disks labeled A, B, ..., O. The A disk is the primary disk which contains the boot blocks and must always be there. The disks are broken up into logical allocation units which are 1, 2, 4, 8, or 16 Kilobytes. Space for files is taken in the allocation units. The size of the directory is static, and is determined by constants in the BIOS. The manufacturer of a system decides on the directory and allocation size based on what their typical user would require.



An entry in the directory is 32 bytes. It contains information about the file size and pointers to it. There is no information about the date created, owner, protection and other information which is typical on larger computer systems. More important than not having this information, is that there is no room for any of this. As mentioned earlier, the CP/M file system was designed in the early days of microcomputers. In order to maintain compatibility, the designers were stuck with a file system that is lacking in several areas.

4.2.6 Putting it all together. We will now look at how this all comes together by following the execution of the following program:

```
                program sample;
                type
                     block = packed array[1..128] of byte;
                var
                     buffer : block;
                     file_1 : file of block;
                begin
(1)                 reset(file_1,"B:MYFILE.TXT");
(2)               read(file_1,block);
(3)               while (not eof(myfile) do
(2)                    read(file_1,block)
                end.
```

In this example, 128 byte blocks were chosen because that is the size that CP/M works with. It is the responsibility of the compiler to generate the routines for blocking and deblocking if the user wants to read by lines. This is a detail which is not necessary for what we are looking at now.

When a program wishes to open a file, it must first create a File Control Block, or an FCB, containing the name of the file and some additional space that is used by CP/M while accessing the file. The FCB is generated by the compiler in its open (reset or rewrite) subroutines. which then call the BDOS with the OPEN function.

When BDOS gets the open call, it examines the file name, and sees that the file is on the B disk. A select disk call is made to the BIOS which returns the disk descriptor table. The BDOS examines this to find the size of the disk, the size of the directory, and where the directory starts. It then issues the HOME call to the BIOS. Once this is done, all disk I/O (as long as another disk is not read from in the meantime) is done with the following sequence of BIOS calls: SETTRK, SETSEC, SETDMA, and READ.

After the select disk call, the BDOS now knows where the directory starts, and begins reading it sequentially, searching for the entry of the file MYFILE.TXT. Once found, the pointers to the blocks occupied by the file are copied into the FCB. A return code is returned to the application program indicating whether the file is found. How this is interpreted is dependent on the compiler.

Each READ statement is translated into a read sequential call to the BDOS. When BDOS gets the call, it looks in the FCB to determine the next block to be read. It then calls the BIOS as in the above mentioned sequence to perform the read. The BDOS can determine when the end of file is reached when it examines the FCB. The EOF return code must be saved by the routines generated by the compiler so that statement (3) will execute properly.

4.3 CP/Net: The Microcomputer Network

Now that we have the background on how the operating systems is organized, we will look at how these machines can be put together to form a network for sharing data and devices. There is really only one microcomputer networking system currently available: Digital Research's CP/Net. This is the only software sold which is adaptable to a wide range of hardware configurations. There are other systems, but these are sold by the manufacturers of particular microcomputers, and are usually just modifications to the BIOS to allow for networking. In principle, they all use the

same scheme, although the implementation may be different.

4.3.1 CP/Net. Digital Research developed CP/Net for the same market as CP/M: Any 8080, 8085, or Z80 based system. They put the hardware dependent parts into one section, leaving the manufacturers of machines to write only this one small part, thus it is easy for a manufacturer of a machine to have a network with their machines.

CP/Net runs with workstations and fileservers. The fileservers control disks and printers which are shared by the workstations. The file server must run MP/M, which is a multi-tasking version of CP/M, and the workstations must run CP/M. There may be up to 16 workstations logged onto a file server at a time. The workstations are regular CP/M machines, although they may be diskless. The workstations with disks must have previously booted CP/M (from disk) before loading CP/Net. The diskless stations must load everything from the network.

There is a command, NETWORK which allows the user at the workstation to assign a logical disk name to a physical disk on a file server. For example, the command NETWORK B:=D:[2] issued at a workstation, causes its logical B disk to be the physical D disk of server 2. The user has great flexibility in accessing many different files on different servers. The NETWORK command can also be used to assign the LST: device to be any printer on any server. If printer 2 on server 3 is wanted as the LST: device, the command NETWORK LST:[2,3] is issued.

Before a user can access any files on the network (s)he must log on to a file server on the network. Once logged on, (s)he has unlimited access to any of the services of that server. It is possible to be logged onto several or all file servers simultaneously.

The last additional feature offered by CP/Net is a "mail" service. A user may send a message to any station logged onto the same server, and the user at the receiving node can then issue a command to receive the message.

4.3.2 CP/Net Implementation: Workstation. CP/Net does not care how the network communication is achieved, as long as there is a logical point-to-point link between each workstation and server. A logical point-to-point link may be physically implemented as a bus, but since the stations communicate only with servers, it appears to be point to point.

There are two versions of CP/Net: The version for diskless workstations, and the version for regular CP/M machines. We will first look at how the regular version is organized, then how it is handled for the diskless workstations.

CP/Net is loaded into a machine running CP/M when the user issues the CP/M command CPNETLDR. This command loads the various sections of CP/Net into the memory of the workstation.

The first section loaded is the NIOS or Network I/O section. This is loaded just below the BDOS, and it overwrites the CCP. The NIOS is the part that contains the hardware dependencies for

communication, just as the BIOS contains those for the disk in CP/M. The next section loaded is the NDOS or Network Disk Operating System. This is equivalent to the BDOS. Finally a new CCP is loaded to accept commands from the user. The memory map of a system is shown in fig 4.2. Note that the additional memory required by CP/Net is 3.5K bytes which is not much on a 64K byte system. Also note that the size of the NIOS is fixed.

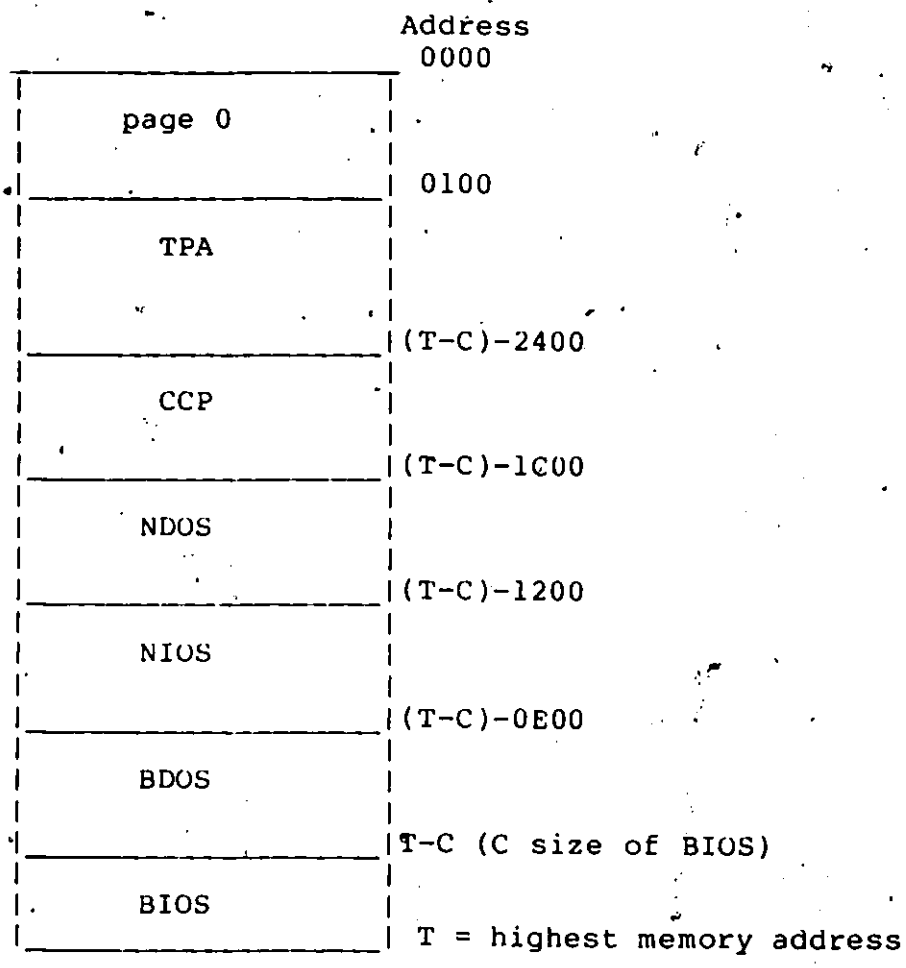


fig 4.2
CP/Net memory organization.

The NIOS is similar to the BIOS in its construction. It starts with a jump table to the various routines. They are basically

routines to send and receive messages, as well as routines for initializing and interrogating the status of the network. There is a warm boot routine which is called each time a warm boot is performed. This has no specific function assigned by DR since the NDOS handles a warm boot entirely. It is available to be used by the designer for anything (s)he may line to add, such as inquiring if there is any mail for the station, or simply interrogating the status of the network.

When the NDOS (Network Disk Operating System) is first initialized, the jump to the BDOS at address 5 is replaced by a jump to the NDOS. In addition, the addresses in the BIOS for the warm boot, and console and printer I/O are replaced by jumps into the NDOS. Each call to the BDOS by an application program is now trapped first by the NDOS. When an application program terminates with a warm boot, a warm boot is not performed by the BIOS as in standard CP/M. The NDOS reads in a file, CCP.SPR, from the A disk of the workstation. This differs significantly from CP/M in which both the CCP and BDOS are read in. This is an important point to remember: The implications are that the NDOS, NIOS, and BDOS cannot be overwritten by an application program as in standard CP/M.

The NDOS maintains a table which contains the mapping of the logical to physical network disk drives as well as indicating which (if any) drives are local to the workstation. The NDOS also gives the application program several additional system calls over and above what the BDOS does. They are essentially

the calls for the additional commands: login/logout, NETWORK, and send and receive mail. They allow application programs to issue any of the network commands. These can be useful in an office environment where a user interface may be written to "hide" the computer system from the end users.

The CCP which is loaded is essentially the same as the CCP that comes with standard CP/M, however, there are some slight modifications. One of the modifications introduces a new bug. Under standard CP/M, typing ^P in response to CCP's prompt causes all console output to be written on the printer as well. This continues until a warm boot occurs or a second ^P is typed. Under CP/Net, typing the first ^P results in a message telling the user that the printing is going on. This is necessary since it is being spooled and the user does not have the sound of the printer to confirm that (s)he did indeed type the ^P. The spooling continues until a second ^P is struck. A warm boot does not terminate printing as in standard CP/M. Later we will see a serious bug in the way that CP/Net handles this.

The version of CP/Net available for diskless workstations, is essentially the same. The workstation must be a microcomputer (8080, 8085, or Z-80) some RAM, and a console There must be some sort of a network boot which initially loads the workstation with the BIOS, BDOS, NIOS, and NDOS. A warm boot is then performed by the NDOS which loads the CCP, The file CCP.SPR must reside on the logical A disk which is on the server. In the case of diskless workstations, there must be at least one server that does not require a user to log on before using its services.

Lets now look at the flow of requests by an application program. We will first examine how requests for serial devices are handled, and then the local and network disk devices.

The serial devices we are looking at are the console and list devices. Remember that the jump to BDOS at address 5 was replaced by a jump to the NDOS. When a call is made to read or write a byte at the console, the byte to be written is put in register E, and the write console function code (02) is loaded into register C then a call is made to address 5. Control is then passed to the NDOS which examines register C and sees that it is not a disk operation, so the NDOS passes the request to the BDOS. The BDOS then calls the BIOS to actually perform the function. Now, the address in the BIOS of the console routines was replaced by a jump to the NDOS which examines the request to see if it should be routed over the network or not. If not it calls the BIOS routine to read or write the byte. This complicated process is done to reduce the size of the NDOS. Since there are quite a few types of calls for serial I/O, it isn't necessary to duplicate the services of the BDOS.

Lets look at a serious bug in CP/Net due to this complicated process. Suppose that the LST: device is on the file server. Now suppose that we type ^P to have our console output printed as well as being on the screen. The NDOS traps the ^P and starts buffering each byte which is supposed to go to LST:. When a buffer is full, it sends it to the server. When a second ^P is typed, the NDOS flushes its buffer, and sends it and a close

spool message to the server, which can start printing the file when the printer is free. This process almost works. The problem occurs because of the BDOS which performs a similar function under standard CP/M. If the BDOS somehow traps the ^P instead of the NDOS, it will start sending the console output to the printer as well. The BDOS can trap it if the user types the ^P before the prompt for input appears. Depending on how, when, and how many times it is done, there can be duplication of each byte sent, or the user is unable to stop spooling. When the second ^P is typed, the NDOS may trap it and start spooling unaware that the BDOS is performing the same function. The user does not know by looking at the screen if it is being spooled or not, and this can create quite a confusing situation as well as filling up spool space quickly.

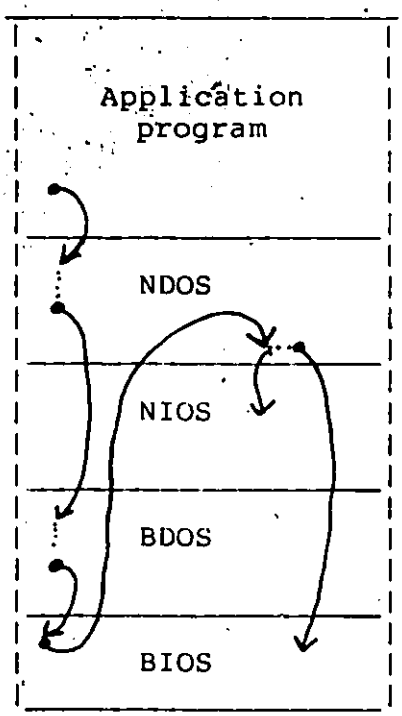


fig 4.3
Control Flow for Serial I/O request.

Now lets look at how the requests for disk I/O are handled. When the program makes calls to the BDOS, they are trapped by the NDOS. The NDOS looks in register C and sees that it is a disk function. It then looks at register DE which point to the FCB. The first byte of the FCB is the drive name on which the file resides. The NDOS then examines its table to determine if the disk is on the network, or if it is local. If it is a non-network request, it is passed directly to the BDOS and handled normally, otherwise, the request is put into a message packet and passed to the NIOS which transmits the request to the file server. The file server then processes the request and sends back the result.

Using the same sample program in section 4.2.6, lets look at the flow of requests from the program. First we will assume that before running the program, the user has issued the command NETWORK B:=C:.

When the open request is made for the file, the NDOS looks at the FCB and sees that it is on the B disk. It then looks in its table and sees that the B disk is actually the C disk on the server. It copies the FCB into a message packet replacing the B disk name by C. The packet also contains the open function code. The packet is then passed to the NIOS which sends it to the server. When the server receives the packet, it looks at the function code and sees that it is an open. It tries to open the file which the FCB describes. If successful, the fileserver then copies the FCB into a table, and returns its address in the table

along with the return code from the open. Meanwhile, at the workstation, the NDOS calls the NIOS to receive a message. Once received, control returns to the NDOS which saves the returned address in the program's copy of the FCB, then returns control to the application program passing it the return code from the operation.

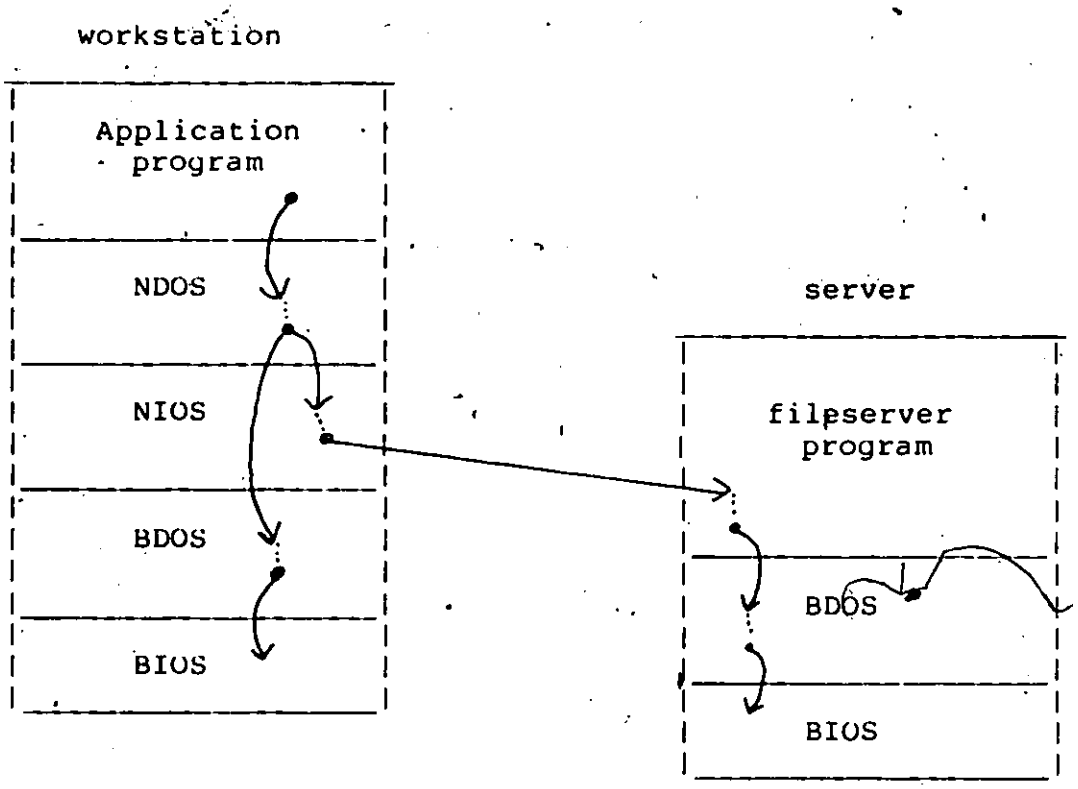


fig 4.4
Flow of control for disk request.

When the read call is issued, the NDOS examines the FCB and sees that it is a network disk. No checks are made to see if the file has already been opened, rather it just takes the address returned by the server, packages that along with the relative block number (not a physical disk address) of the file to be read and sends it to the server. The server then uses that address to find the FCB in its table, and issues a read. No checking is

done here either. The server returns the block which was read and the return code to the workstation. The NDOS then copies it to the program's buffer.

Notice in figure 4.4 how this flow of control follows a nice logical route as opposed to the serial I/O portion of CP/Net. The fileserver provides an extension to the BDOS and BIOS of the workstation.

4.3.3 CP/Net Implementation: Server. As mentioned earlier, the file server must be a microcomputer running MP/M. This requirement makes sense logically. It is much easier to write a server for a single workstation and run this task many times to handle other stations. The problems of having efficient response time is then passed to the operating system. Digital Research has given this problem to manufacturers of machines since the response time of MP/M depends on how much overlap of I/O and processing that can be provided in the BIOS. Usually very little can.

There are three parts to the server: queue management, network interface, and server. The queue management is a function of normal MP/M. The operating system will queue and de-queue a packet in an non-interruptable state, so that these packets are available to all tasks as a means of inter-task communication. When a task wishes to send a packet to another task, it issues a system call which identifies the queue, the packet, and the task for which it is intended. When a task wishes to remove a packet

from a queue, it issues another system call with the name of the queue. The system returns the first packet from the queue. Two queues are required by the CP/Net server: a request queue and a response queue.

There is one re-entrant network interface task for each workstation that can be logged onto the server. Each task handles its communication link to the workstation. (Logically point-to point). When a request arrives, it disassembles the packet, and issues a system call to place this packet on the request queue. It monitors the response queue, and whenever there is a packet, it removes it from the queue, and start sending it back to the requesting workstation.

There is one actual server task. This task does nothing but remove requests from the request queue, perform the service and place the result on the response queue.

Conceptually the file server software is very simple and clean. The fact that there are many tasks to do the work make it easy to maintain and allow for a certain amount of overlap. For example, several stations can be making requests simultaneously while another is getting a response, and the server is servicing a request.

One major problem with the file server is the operating system: MP/M is not a very popular operating system. It is usually better to use several microcomputers than to have a multi-user microcomputer. There is very little application software available that takes advantage of the multi-tasking features of MP/M.

It is also very difficult for manufacturers to develop a BIOS for it. When MP/M is running on a microcomputer, it is often very slow and cannot make efficient use of the hardware. This can be expected since the 8080, 8085, and Z-80 CPUs are not really designed for this sort of thing.

4.3.4 CP/Net Problems. In addition to the above mentioned problems, there are other serious problems that make CP/Net unsuitable for use in computer science education at the university level. These problems fall into the following categories:

- Network security,
- Versatility,
- File security,
- Unrestricted use of resources,
- Performance,
- Not standard CP/M.

CP/Net uses a very poor system for logging onto the network. There is one password that must be entered with the logon command. That does not mean one password per user or workstation, but rather one password per network. Needless to say that with students, this is entirely unacceptable. In addition, when diskless workstations are used, this level of security does not even exist. The logon command is a regular program. Since the station is diskless, it must reside at the file server, and in order to load a program, the user must be logged onto the network, so each station must be perpetually logged on.

CP/Net is not very versatile. Because of the organization and flow of requests, it is essential that the file structure on the file server be identical to that on the workstation: The work stations can only run CP/M. Similarly, the file server must be run on a CP/M machine. It would be nice if the file server could be run on another faster (more expensive) machine, while the work stations could be less expensive 8 bit machines, or if a few machines with other operating systems could be added to the network (for sharing data and disk space, not programs). This problem arrives not because CP/M and CP/Net are from DR, but by the way requests are serviced: By file.

File security is another serious problem. Since CP/M keeps no information in its file system as to when files are created or who created them, there is nothing that the file server can do but give total access to all files created by all students. Any student can copy, erase, rename, etc. a file created by any other student. If someone erases the file CCP.SPR, the whole network will crash. When a student at a station has access to a disk on the file server, (s)he has access to the entire disk.

There is another problem associated with sharing files on a disk: there must be some convention followed by everyone on naming files. A file named ASMT1.PAS could be owned by any one. Again, this problem comes from the way the requests are serviced. Since the requests are made by file, and the file structure of CP/M is limited, CP/Net is stuck with no security.

The next problem, unrestricted use of network resources stems from the same cause as the other problems. There is no logon, so CP/Net does not know one user from another. Since each user has unrestricted access to the server's disks, (s)he can fill it with whatever (s)he wants. There is no dates or owner indication on the files, so it is the responsibility of each user to remember what files (s)he saved, and to erase them when not needed. The disk or directory may become full if students forget to erase old files. When this happens, the system manager must somehow free some space on the disk. How does (s)he do it? Any file erased might belong to a student who needs to hand it in the next day.

Unrestricted use of the printer is another problem. This together with the problem of the control P mentioned earlier can spell disaster. If a student does not know that the console is being spooled to the printer, (s)he may work for hours, and in doing so, fill up the disk with the editing session. A full screen editor can generate a large spool file if its output is directed to a printer as well as the screen. Even if the disk does not fill up, a long listing of garbage is wasteful, and ties up the printer when other students wish to use it. If the reader has assumed that CP/Net has no facility for purging a large unwanted print file, (s)he is correct.

Finally, we get to performance. There are several areas that affect the performance of CP/Net. First of all, each time a warm boot is performed, the CCP has to be reloaded. Since diskless stations are used, this would mean that this file would have to be loaded from the file server. Students tend to edit, compile,

link, edit, etc. Each of these processes end with a warm boot. The probability of two or more warm boots happening at about the same time is rather high. This will slow down the response of network services. Since the response is slowed down, it is probable that another user may cause a warm boot to be done, slowing things down even more.

Each time a file (including the CCP) is opened, the server must go to the directory to find it. Designers of mainframes and minis have learned that putting the directory in the middle of the disk reduces the average length of the seek needed to get to the file. Most of the microcomputer industry has not yet learned this and, therefore, the average seek from the directory to the file is half the disk. The seeks can slow the file server down. MP/M does not sort disk requests by order of tracks to reduce seeks.

A more serious problem concerning the directory arises from the file organization. The directory is a sequential file. Each open requires reading this file to find the entry for the file to be opened. Most recently created files tend to be at the end of the directory. Most recently created files tend to be the most recently used. Most recently created files also tend to be close to the center of the disk. Add to this the fact that MP/M does not optimize disk seeks and the problem here is obvious.

Microcomputer operating systems were designed for machines with floppy disks. These can not hold much data, and therefore have a small directory. Winchesters, on the other hand, have a large

capacity, and therefore, need a large directory. CP/Net running with a server with a large winchester can get very slow when the disk starts getting full.

Finally, there is the problem that it is not standard CP/M. Although almost any program that runs under CP/M will also run under CP/Net, there are two reasons that all will not; (1) the NDOS, NIOS, and BDOS cannot be overwritten, and (2) CP/Net returns a different version number than CP/M. There are some programs that may wish to use all memory up to the BIOS. This is fine in standard CP/M since the BDOS and CCP are reloaded. These programs will crash CP/Net workstations when they terminate. As for the second problem, there are many programs that check the version number in their initialization process. They will exit if the version is not correct. This problem is not too serious; a clever programmer can zap the module, usually in an hour.

These problems with CP/Net are much more apparent in the student environment than in an office environment. In such a case, network requests will be few and far between. A typical user there will load a word processor or spreadsheet program, and work on that for quite a while. Printing requests will be done through these application packages and there is little chance running into the problems there. If security of data is a problem in the office, a microcomputer network cannot be used or sensitive data can be kept only on local disks. The office will typically have fewer users per workstations and monitoring of who is doing what is possible.

Chapter V

Design Philosophy

We will now be looking at what we must have in the system so that the system is suited to our needs. There are certain features that are the responsibility of the operating system, and others are that of the network software. In this section, we will look at how CP/M satisfies our needs, and we will look at what features were incorporated into CPN so that it gives us what we want.

5.1 Network and Operating System Requirements. We will now look at feature we are given by CP/M. They are:

- it is well established and much software is available,
- it is simple for the first-time user to start with,
- it is simple and easy to maintain,
- it is easy to improve on.

There is a wealth of software written to run on CP/M. This is shown by the variety of compilers/interpreters we are running: Pascal, COBOL, PL/1, Ada, LISP, Prolog, and C, as well as word processing and spreadsheet software.

The simplicity of CP/M is important for two reasons: There is very little operating system "background" necessary before using the application software, and it is simple enough so that a student can completely understand its workings. For example, in a UNIX type of environment, an understanding of the tree structured directories are required before the student can start working.

In other operating systems, there is a large assortment of "system commands" that a user must worry about. Under CP/M, the student can learn all the system commands in a matter of minutes. This means that (s)he can start learning the material covered in the course sooner. There are only the essential commands: type a file, look at a directory, erase a file, and copy a file.

In a second year introduction to operating system course, the simplicity of CP/M is a benefit to the student. At this point, the student sees the operating system as a provider of services. CP/M provides minimal services such as file system maintenance and I/O interfacing. The simple file system of CP/M can be totally understood by these students. Students see why the various system calls are necessary, and they can understand how it is implemented. In third year, when they are taking a "real" operating systems course, they have a very good background for the basics, and can then better understand the more complex features in these larger systems.

5.1.1Network requirements. Just as CP/M gives us a base on which to run our application software, and the network software gives us a base on which to run CP/M. The network software must give us:

- Steady response time, even under load.
- System security (crash proof).
- Cheat proof. Students cannot copy or "share" assignments.
- Theft proof. Students cannot steal application software.
- the ability to limit resources.
- Flexible. As new application software becomes available, it should be easy to add it.
- Simplicity. It should be easy to maintain, modify, upgrade, etc.

Lets first look at providing a system that has a steady response time. Obviously any system with a single fileserver will have a longer response time when it is heavily used, so we will look at a way to reduce the need to access it.

There are only a few times when the user really has to access the fileserver:

- Logging on.
- Running application software.
- Accessing student's assignment.
- Printing.

Logging on by itself is a minimal use of the network. First of all, it is only done once. Secondly, it all that is required is

the transfer of the userID and password, and possibly a small boot program.

By far, the largest use of network resources is the running of application software. The application software tends to be large, and if it has to be transferred to the workstation at each invocation, this will put a tremendous load on the fileserver. If the student transfers the application software to the workstation when (s)he logs on, the transfer need only be done once.

The largest assignment a student would typically have would be their final COBOL assignment which if heavily commented would be at most 35K bytes, or 100 pages. The average size of an assignment is more like 6K-10K bytes. Being able to transfer this to and from the file server only when necessary, (initially and when finished working), would further reduce the load.

The flexibility requirement is satisfied by having CP/M as the operating system if it is standard CP/M. It should not be some network variation, such as CP/Net, that will not run certain application programs. The system is even more flexible if it allows the workstation to run different operating systems.

In the following sections, we will look at what CPN does to perform well and to be secure at all levels.

5.2 The RAM DISK. Remember from section 4.2.3 how the BIOS is called for disk accesses. The important point for us to note is that the (track, sector) pair specifies a unique 128 byte block.

Normally this is a sector on the disk, but there is nothing that stipulates it has to be. As long as anything written to a given address can later be read from that same address, CP/M does not care how it is stored.

Although the Z80 processor has a 64K byte address space, and CP/M is only able to handle 64K bytes of memory, it is possible to have banked memory which CP/M does not know about. This gives the ability to physically have more than 64K bytes of physical memory. At most the processor can directly access 64K at any one time, but banks can be mapped in and out as needed, giving us access to much more memory.

This additional memory is logically broken up into 128 byte blocks and the BDOS maps the (track, sector) pair to a block of memory. To the BIOS, this appears simply as a disk. To the user, it appears to be a very fast disk.

5.2.1 RAM Disk Benefits. We get a lot of advantages in using RAM disks over both floppy disks and no disks. As a matter of fact we get the advantages of both, the disadvantages of neither, plus an additional advantage, speed.

First of all, we have the advantages of diskless workstations as mentioned in section 3.3

- greater reliability (no mechanical breakdowns).
- Students cannot steal software
- Students cannot load their own software.

We also have the advantages of a station with disks: We can

transfer the application software to the workstation to reduce the load on the fileserver. The typical user then transfers a 70K byte system once, transfers his/her 8K byte program twice (from and back to the B: disk), prints the 14K byte listing once. The total network transfer for a typical session is 100K bytes. These are only transferred when necessary.

An additional benefit of the RAM disk is that it is much faster than a floppy. Everyone appreciates a fast response time. For example, it takes 8 seconds to load Wordstar into memory on a DY-4 with 5" floppies. It takes less than a second to load it from a RAM disk.

5.3 Network Independent boots. Student programs are far from bug free. It is very common for a student's program to get into an endless loop. This requires the machine to be reset and a boot to be performed again. There is really no need for this reboot to access the network, the student has already logged on.

If, when the student logs on, we transfer boot information to an area on the RAM disk, a network independent boot can be performed. When a reboot is performed, this information must be transferred from that area of the RAM disk. This kind of reboot is fast since it just involves a memory to memory transfer. The boot information must be removed when the student logs off to prevent anyone else from using his/her account.

In order to have this network independent boot, the boot EPROM of the workstation must be modified to give the student a choice:

booting from the network, ie. logging on, or the network independent boot.

In looking at what we gain by providing this, we get:

- reduced traffic on network,
- a fast boot (good from the student's point of view),
- network independence (if fileserver crashes).

The cost of providing this is the cost of saving a copy of the operating system in a bank of memory. This comes to 8K bytes out of a 256K RAM card.

5.4 Log on process. In looking at ways to reduce the load on the fileserver, let us look at the time when there is the greatest load: logon. When a student is logging on, we know that they will also want to transfer the application software to the workstation. This ranges from a small system such as PROLOG (52K) to a large one (256K) like ADA. Even with the small system, the transfer of 52K bytes is much larger than the 8K byte transfer of the student's program from his/her area on the fileserver.

Lets now look at what is putting the load on the fileserver. The speed of communication is 800K baud, or about 100K bytes a second. This does not appear to be the bottleneck; if we could transfer the 256K byte system in 2.6 seconds, we wouldn't worry about a load on the fileserver since the time for loading a system is so short. This is not the case.

We cannot even transfer the 256K bytes from the Winchester to RAM on the fileserver in 2.6 seconds. The bottleneck is the disk.

When performing disk accesses, there are several operations that take time:

- seeking the track (average time 36 milisec)
- searching the track for sector (average time 4.5 milisec)
- reading a block (512 bytes) of data (0.9 milisec)

First of all , notice that the physical block size on the disk is 512 bytes. CP/M, however, works in terms of 128 byte sectors, and it is the responsibility of the BIOS to perform the deblocking and to return 128 byte blocks to the BDOS. The block is actually read once, for every 4 BDOS calls, if this file is being read sequentially. The only overhead in reading 128 byte blocks is the execution of the additional instructions for deblocking if the physical block remains in memory. When a single user is accessing the fileserver the block does remain in memory, but when there are several users accessing the fileserver simultaneously, the block may not remain in memory. It is highly probable that the same block is read in 4 times.

In examining the time required for a disk access, we notice that most of the time is spent looking for the block of data. If we can insure that the file is contiguous (and we can with CP/M), that means that once we start reading the file, the heads are positioned, thereby eliminating the seek from subsequent reads. The initial block read will take the full time to transfer, the rest will go fast. The average time to read the first block is about 42 milisec. If we immediately read the following block, it will take an additional 2 milisec. Thus we get twice as much data read with an increase in time of less than 5%. Again, this

is true if only a single user is accessing the fileserver. If several users are accessing the fileserver, there may be seeks to each other's files greatly increasing the time to access the file.

Finally, there is a certain amount of overhead in the transmission of each packet; handshaking, the acknowledgment, etc. It takes far less time to transfer one packet of 512 bytes than to transfer four packets of 128 bytes.

Looking at these facts, it seems logical that we should use blocks larger than 128 bytes when downloading the application software. using a block size of 512 bytes prevents additional reads of the same block, and using a still larger size prevents seeks from other users from slowing us down. What should be the limit on the block size? The CRC used with the Zilog SIO chip is less effective with blocks larger than 2K, and that is why we chose a packet size of 2K.

Lets now examine what we gain by using 2K blocks. Consider table 5.1. When there is no other user on the system, it takes about 3 times as long to load a system using 128 byte blocks than it does using 2K byte blocks. This is the overhead incurred by having to send 16 times as many packets. For a single user on the system, this time is noticeable, especially when loading a larger system such as COBOL. From the student's point of view it is not the 3 times faster that makes a difference, but the absolute time saved: 48 seconds. And that is when no one else is using the network.

Now consider the case when there is only one other user loading a different system. The loading of a different system will force seeks in between each request. The time for the student has more than tripled, while in the case of the 2K byte blocks, there is only an increase of about 70%. Again, looking at the case of COBOL in terms of absolute time, a single other user accessing the network will cause an increase of 2 minutes! This is the type of response we get using CP/Net, and it is not acceptable.

128 byte blocks

	no load	1 station accessing other file.
PASCAL (72K)	20 seconds	66 seconds
COBOL (180K)	62 seconds	143 seconds

2K byte blocks

PASCAL (72K)	7 seconds	12 seconds
COBOL (180K)	16 seconds	23 seconds

Table 5.1
128 byte vs. 2K byte transfer times

What size packet size should we use for the other disk transfers? First of all, when a student is logging on, we know that there will be a large file transfer, and we know that this will be sequential. When the student is printing or accessing his/her private files, we have no idea of the size of the file, but they are usually small.

The time (no load) required to transfer the average file (8K bytes) from the student's area to the fileserver to the work-

station is 2 seconds. We won't gain much in absolute time by optimizing here. Even if we increase the efficiency by a factor of four, we will only save 1.5 seconds of the student's time. At a very busy time, we might be able to save the student 4 or 5 seconds. Accesses to the student's disk is not always sequential. CP/M uses an allocation on the disk which tend to have files scattered all over the disk after the creation/deletion of many files. The use of larger blocks in only beneficial if the accesses are sequential.

Since printing is done by transferring what is on the screen to the fileserver, the transfer time here is limited by the speed of writing on the screen: about 4800 baud. In both of these cases (printing and file access), we gain very little by having blocks larger than 128 bytes, and it complicates the rest of the software. The benefits are not worth the cost.

5.5 Fileserver Network Although it is nice to have all the workstations and all fileservers connected on a single network, this would mean that all traffic would pass over a single wire. There would be quite a bit of contention for that wire with 80 stations trying to access 5 fileservers. By having 5 seperate networks, the fileservers can essentially work in parallel.

The only reason for wanting a single network is so that someone can sit down at <u>any</u> station to access his/her files. Each of the fileservers can have copies of the application software and can do its own printing.

Remember that the students' files are small compared to the loading of a language system. The average time to get a student file from the private disk is 2 seconds. even if it takes twice as long the wait is not that bad. On the other hand, a large language system such as ADA (256K) takes 26 seconds to load. Any increase here is unacceptable.

We are willing to allow for a decrease in performance when accessing the other fileserver, and we do not want this feature to cause a decrease in performance in logging on and printing. It makes sense to have seperate network connecting the fileservers.

This fileserver network is smaller than the workstation network; it only has five nodes: the fileservers. It is feasible to use a more expensive (better performance) network interface board as well.

As it turns out in our case, there is a 50% increase in the time it takes to access a file on another fileserver when there is no load. This 50% increase in time remains constant relative to the time it would take on a busy network. For instance, if the network is so busy that it takes 8 seconds to perform a local B: disk transfer, it will take about 12 seconds to do an interfileserver transfer. This is assuming that the load on the two networks is equal, which is usually the case.

5.6 Virtual disks. One of the problems with a network such as CP/Net is that it is based on files: an open request at the workstation translates to an open request at the fileserver, a read request at the workstation gives a read request at the fileserver, etc. This is fine if there must be sharing of files, but we do not want sharing of files.

In section 4.2.4, we gave problems with a file based fileserver. Reviewing these again we have:

- performance (due to many opens and a sequential directory)
- security (microcomputer file system has no security)
- unrestricted use of resources.

If we wish to have file security, we must maintain our own directory over and above what CP/M has. In it we must include the owner of the file. In addition to making the fileserver software more complex, this also decreases performance: we have two levels of directory to search through. We also have to maintain the integrity of the two seperate directories and make sure that they agree.

If we wish to limit the amount of disk space a student uses we have more problems. We have to maintain records of which blocks are used by the student, and every write operation must be checked to determine if it is rewriting an already allocated block (which we don't care about) or writing (allocating) a new one (which we do care about). This gets very messy since we must duplicate much of CP/M's file maintenance.

The concept of virtual disks was first used by IBM with their VM operating system. The idea behind virtual disks is that a virtual machine sees its disks as a starting at cylinder zero, and going up to cylinder n-1. In reality, the virtual disk is a collection of n contiguous cylinders on a real disk. When accessing the disk, a certain offset is added to the virtual cylinder address to arrive at the real cylinder address, and the operation is performed. Security is simple: if a virtual machine has n cylinders allocated to it, all that has to be done is to check that the requested cylinder address is less than n-1.

We can use a similar concept in our system. Remember in our discussions of the RAM disk (section 5.2) we told how when the BDOS calls the BIOS to perform a disk operation, all it cares about is that a given (track, sector) pair references a unique block of data. We can translate the pair into a single number:

```
block_number = (track * sectors_per_track + sector)
```

Each student is allocated a file (virtual disk) containing n blocks (virtual sectors). The block number that we compute will be used to reference a block within that file. It is simple to enforce security rules and to limit the space a student uses.

This file of n blocks is the only file that a given student has read/write access to. It is simple to maintain a list of size one. We may also allow for several others that everyone has read only access to. This again is simple to manage.

In order to limit resource usage (prevent a student from taking up too much space on the fileserver), we need only check that for

every write operation, that the block number is less than the number of blocks allocated for that student. Again, this is simple to do.

Since students only see the virtual disks, they are not even aware of the real files that are on the fileserver. They cannot erase important files, issue a command to cause the system to crash, etc. Under CP/Net, it is easy to crash the system by erasing important files.

5.6.1 Virtual Disk Implementation. In order to implement this, the network accesses reside in the BIOS in the disk routines. Whenever there is a request for a disk, it is first checked if it is for the RAM disk. If not, the (track, sector) pair is translated to a single number, and the request is sent to the fileserver (along with the data if it is a write).

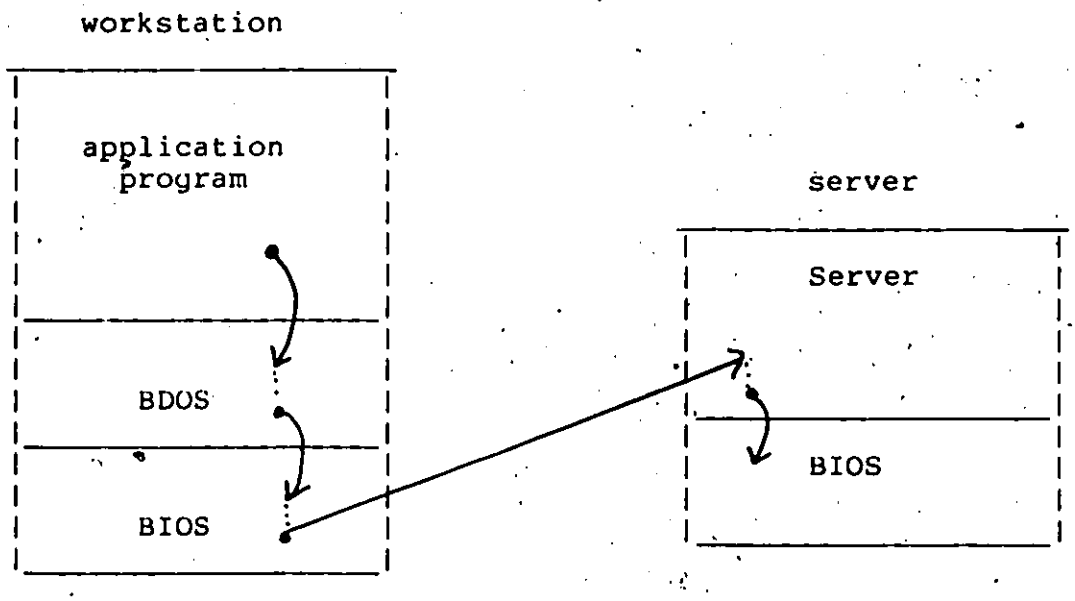


fig 5.1
Logical flow of control for disk request.

5.6.2 Virtual Disk Advantages. The advantages of using virtual disks instead of a file based fileserver are numerous:

- security easily maintained,
- resource usage can be limited,
- increased performance,
- workstation's CP/M is standard,
- independence of workstation's operating system.

The first two points were discussed above, now lets look at the others.

Performance is better since we do not have to perform an open before every file access. The student's virtual disk need only be opened once when he/she logs on. Remember that the opens required on a file based system cause seeks to the beginning of the disk, and a sequential search through the directory. We also increase performance by reducing the time for the few opens that we do have to perform. Since the virtual disks tend to be large (around 100K), we will have less files in the directory than if we would have many small files on the disk. Since the directory is smaller, the sequential search through it is faster.

When running CP/Net, we have something that sits above the BDOS (the NIOS and NDOS), and for some application programs, this does not appear to be a true CP/M machine. When using virtual disks, the network portion stays in the BIOS, and the machine has the standard BDOS and CCP. It is <u>standard</u> CP/M.

Finally, there is nothing about a system like this that ties the operating system of the fileserver and the workstation together.

The fileserver does not care what the user does with the virtual disk. The workstation could run UCSD, which would maintain its type of file system on a virtual disk. A workstation could even be a 16 bit machine running MS-DOS. Each different operating system would require a different virtual disk, (since the file structure is different) but the server would never know the difference.

This works in the other direction as well. This type of system makes the fileserver independent of the workstations. We may decide to upgrade our fileserver to a higher performance machine. The fileserver program can be written in a high level language and could be hardware independent.

5.7 Printing. Finally, we look at printing services. The easiest place to capture things for printing is in the BIOS in the console and list output routines. We have to be able to turn printing on and off, as well as selecting console or list.

We use the console output for getting a listing combining the compile, link and run of a program; the list output is needed when we want the output of a word processor, and don't want to have the operating system prompts that we get from the console.

In order to implement these functions, an additional BIOS entry point is added. This call has a parameter indicating whether it is a request for start of console spool, start of list spool, close spooling, or break spooling. Using an additional BIOS entry point will make application software which controls spool-

ing independent on the memory size of the system that is running.

5.8 Concluding Remarks Lets now review what we have, and what it gives to us. Remember that we want:

- Steady response time, even under load.
- Theft proof. Students cannot steal application software.
- System security (crash proof).
- Cheat proof. Students cannot copy or "share" assignments.
- the ability to limit resources.
- Flexible. As new application software becomes available, it should be easy to add it.
- Simplicity. It should be easy to maintain, modify, upgrade, etc.

A steady response time is achieved by minimizing network access, which is done by having a bootable RAM disk which is loaded by sending large packets. The diskless workstation assures that the software is theft proof. Using virtual disks and having all of the network parts residing in the BIOS give us the rest of our requirements.

Chapter VI

Software Details

In this chapter, we will look at the fileserver program in quite some details. We will see that it is really a simple program to understand modify update, etc. The fileserver is written almost entirely in Pascal. Pascal/MT+ is the compiler used. The reasons for that we had. It produces reasonably efficient code, and it allows for externally compiled routines. Some of the routines are written in assembler. The reason for this in many cases is efficiency (for some often called routines), and in other cases (network communication) it is because timing is critical.

6.1 Workstation Requests. There are 6 types of requests that the fileserver will get from a workstation. These are:

- logon,
- open file,
- load system,
- read sector,
- write sector,
- spool.

Most of these should be obvious from their name, but we will

still briefly look at them.

The logon request contains the userID and password which were entered in response to the logon prompt. The fileserver responds indicating success or failure. If successful, additional information is returned with the response: the user's name, name and size of the virtual disk, and several other fields from the entry of the password file.

Since the virtual disks are files, they must be opened before being read. That is the reason for the open request. The open returns a code indicating success or failure, as well as a unique number which is used in further references to the file.

The load system request asks for the next 2K byte block of an opened file. This operation is performed sequentially, and these files are always read only. Either the next block, an error message, or an EOF signal are returned.

The read sector request is made for reading a random 128 byte block from file (a virtual disk). Again, either the data or an error code is returned. The write sector performs a similar function for writing a random 128 byte block to the user's virtual disk.

The spool request may have several subfunctions. It may be a request for status, a request to start spooling, request to stop, or a request to kill spooling. The request is processed and either a success or failure are returned.

6.2 The fileserver. At this time, we will just mention that the network access mechanism is polling. We will be going into more detail in the next chapter but for now lets just assume that we can poll a station and the poll is either accepted or not. If it is not accepted, that indicates to us that the workstation has no need of the fileserver. We can now look at the main loop of the fileserver. It is:

```
initialize;
do forever begin
    for i := 1 to number_of_stations do
        begin
        request := poll(i)
        if poll accepted then
            do case of message_type
                begin
                logon: . . ;
                open: . . . ;
                load_system: . . . ;
                read: . . . ;
                write: . . . ;
                spool: . . . ;
                end case;
        if other_server_has_request then service it;
        if console_command_entered then execute it;
        if printer ready for data then print byte;
    end for;
end do forever;
```

We simply poll each station and process its request. There is nothing complicated about this. We will now look in detail at the way that the various messages are handled.

A	name	std. num	ACNT	PWD	P	SF	DL	DS	CRS

A - Account is active or not.
ACNT - Account number.
PWD - Password.
P - Privilege flag.
SF - Number of copies allowed to be printed.
DL - Virtual disk location.
DS - Virtual disk size.
CRS - List of courses and lab sections.

Acoount record.

NID	^SFCB	LF	AR	^ROL	HS	VFCB	OSIX

NID - Workstation's network address.
SFCB - Pointer to current spool FCB.
LF - Logon flag. Indicates if a logon has been done.
AR - Copy of the user's account record.
ROL - Pointer to circular list of Read Only FCBs.
HS - Fileserver containing virtual disk.
VFCB - FCB of user's virtual disk.
OSIX - Other servers index from open.

User record.

fig 6.1
Account and user record format

6.2.1 Logon As just mentioned, this request contains the userID and password entered by the user. The userID is a letter followed by three hex digits. The letter indicates the colour of the network on which the user has the account. The fileserver has copies of the accounts from the other networks of the same

year for inter-net logons. The hex digits are used as an index into the account file. We then have a letter telling us what file to read and an index into the file.

Reading the account file gives us the account record for that student. Figure 6.1 shows the format of the account record. If the password matches, the logon is performed. A user record is then set up. This indicates whether the user is privileged, the name and size of his virtual disk, the fileserver on which his virtual disk resides, and the user's real name. This information is used in determining what files the user has access to in later transactions.

6.2.2 Open requests The open request contains the name of the file to be opened. The file to be opened will either be user's virtual disk, or for a system (language) virtual disk.

If the request is for the user's virtual disk and it exists on this fileserver, the FCB is found in the user record. It is opened and a flag is set indicating that it has been opened. The flag is used so that subsequent calls will not require an actual open to be performed. A code of 1 is returned which is used with the read and write requests for the file.

If the request is for any other file, we must check if the user has privilege. A privileged user can access anything, while a non-privileged user can only open a system disk. System disks are indicated with a filetype of VLD for Virtual Language Disks. Each user record points to a circular list of 5 FCBs that the

user has (or had) opened. If the file is not on the list, it is added and opened. The least recently used file is dropped from the list. A number between 2 and $2^{15}-1$ is returned to the user. This number is the index into the file plus the number of rotates of the circular list. A record is kept at the fileserver of the number of times a file has been dropped from the list. The difference between this number and the number returned to the user indicates its position in the list. For a non-privileged user, this is not necessary since they will never open more than 5 files, however, a privileged user may be opening many files since (s)he can access any virtual disk. It is necessary to use a system such as this to make sure that the user is not trying to access an FCB which has been rotated off of the list.

If the user's virtual disk is on another server, the request is sent to the other server. The number returned is kept in the OSIX field of the user's user record. We will cover the way the other server handles this shortly.

6.2.3 Load System. The request contains the number returned by the open. The fileserver determines if this FCB is still on the list, and if so, performs 16 reads. This large block is then returned to the workstation. The routine to do this is one of those written in assembler. It is quite simple to do this in assembler, and it can be done much more efficiently than in Pascal.

If the index in the request indicates that the file has been rotated out of the list, an error code is returned. This situation should never occur unless a student is attempting to do his/her own network accesses, or if the fileserver was brought down and came up quickly while a system was being loaded. In any case, it is the responsibility of the program at the workstation to recover from this error by re-opening the file and starting to read it from the beginning.

6.2.4 Read and Write routines. These are similar in the way they are done. The received request has the FCB index, a relative record number (which sector to read or write), and in the case of a write, 128 bytes of data. If the FCB index is 1 it indicates that it is for the student's private virtual disk. If the request is a write, the record number to be written is checked to see if it is in the range of the file (otherwise students could enlarge the size of their disk). Assuming these checks were successful, the read or write operation is performed and, in the case of the read, the data is returned.

If the FCB number is not 1, it is for a system language disk. For a non-privileged user, it must be a read and not a write. The position of the FCB in the list is checked the same way it was for the large (load system) read. If the FCB is found in the list, the read (or possibly write for a privileged user) is performed.

If the FCB index is 1 and the user's disk is on another server, the request is sent with to the other server with the FCB index

replaced by the number in the OSIX field of the user record. The response is simply returned to the workstation.

6.2.5 Spool We have several types of subfunctions that this may be:

- start spooling,
- middle spool record,
- end spooling,
- break spooling,
- spool status,
- kill spool file.

First of all, lets look at the spool data structures. There is a list of free spool FCBs. When there is a request to start spooling, an FCB is removed from this list and a file is created on the fileserver's RAM disk. The RAM disk is used because of its speed. A flag in the user record indicating that the user is spooling is set, and a pointer to the spool FCB is initialized. If there are no free FCBs in the list, an error is returned to the user. This has never occurred in practice, although it is possible to cause it to happen by turning off the printer and printing a lot of short files.

A data record for spooling contains 128 bytes of data. When we get this, we check if the user is currently spooling. If not it is an error, otherwise, we find the user's spool FCB from the user record, and write the next block of the file.

The end spooling record causes the fileserver to close the spool file, and add the FCB to the end of a queue of FCBs waiting to be printed. The spool flag in user record is cleared. The printing of files is done first come first serve.

The break spooling command allows the user to change his mind and kill a file currently being spooled. This simply causes the spool file to be erased and the FCB is added to the list of available spool FCBs.

Upon receipt of a spool status request, the queue of spool files is "traversed", and the owner, size, position, and spool ID (a unique number for each spool file) is returned. It should be mentioned that the spool FCBs are not simply a standard CP/M FCB. They contains a few additional fields for our use (pointers to next, owner, ID, etc).

Finally, the user can kill (purge) one of his files waiting to be printed or being printed. This request contains the spool ID, which the user previously found using the spool status command. The request is verified to see if it is indeed his/her spool file, and if so, it is removed from the queue, erased, and the FCB is returned to the list of available spool FCBs. In the case of the file currently being printed, a flag is set so that the despool routine will treat it as the completion of printing.

6.2.6 Other Server Request There are only three types of requests that are passed between fileservers: open, read, and write. Since the fileserver network is not accessible directly

by the students, we depend on the local fileserver to perform the access restrictions, and requests from another fileserver are not checked. For simplicity, even privileged users can only access their B: disk. If the privileged user wants to use other disks on another network, (s)he must log onto that network.

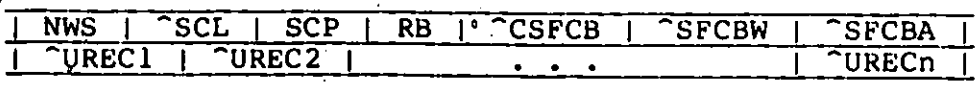
First of all, we will mention that when we send a request to another fileserver, we may get a request from another (or the same fileserver) while waiting for the response. It is possible to get into a deadlock situation if we simply wait for the response. Consider the case when two fileservers simultaneously send a request to each other. If they each simply send the request and wait for the response, they would both wait forever.

To prevent deadlocks, we continuously check for requests from other fileservers while waiting for a response. If we get a request, we service it then continue waiting/checking.

Now lets look at the case of several fileservers simultaneously sending a request to a single fileserver. The hardware being used will buffer of up to three requests. Since users are forced to use workstations of their own year, the most that could be sent to a signal fileserver is two (in the first year lab). A privileged user could intentionally crash a fileserver by forcing all fileservers to access a single one at the same time. The privileged user could crash the system in many other ways as well, and this is not considered to be a problem since privileged users are assumed to be responsible people.

When we receive a request from another server, they are handled quite similar to the way that local requests are handled. The open causes the FCB to be added to a circular list, and a unique number is returned. The read/write routines use this number to identify the FCB. It is quite possible for an FCB to have been rotated off the list. This could happen if a user is logged on for a long time, while many other inter-network users have logged on for a short time. It does not pose a problem since the BIOS at the workstation is designed in such a way that if a read or write request is not successful, an open is issued and a retry is performed. The only effect that the user will see is a delay of about 1 second.

6.3 Fileserver Data Structures. Lets now look at the data structures that the fileserver maintains. The pointer block is the starting place for everything. During debugging, dumps were produced of the data area, and the pointer block was very useful.



NWS - Number of workstations.
SCL - Circular FCB list for other fileservers.
SCP - Station currently being polled.
RB - Request buffer holding request from the station.
CSFCB - Current spool FCB being printed.
SFCBW - Spool FCBs waiting to be printed.
SFCBA - Available spool FCBs.

fig. 6.2
Pointer block.

Consider figure 6.3 in which we show the entire data structures on a network having four workstations. Station 3 is currently

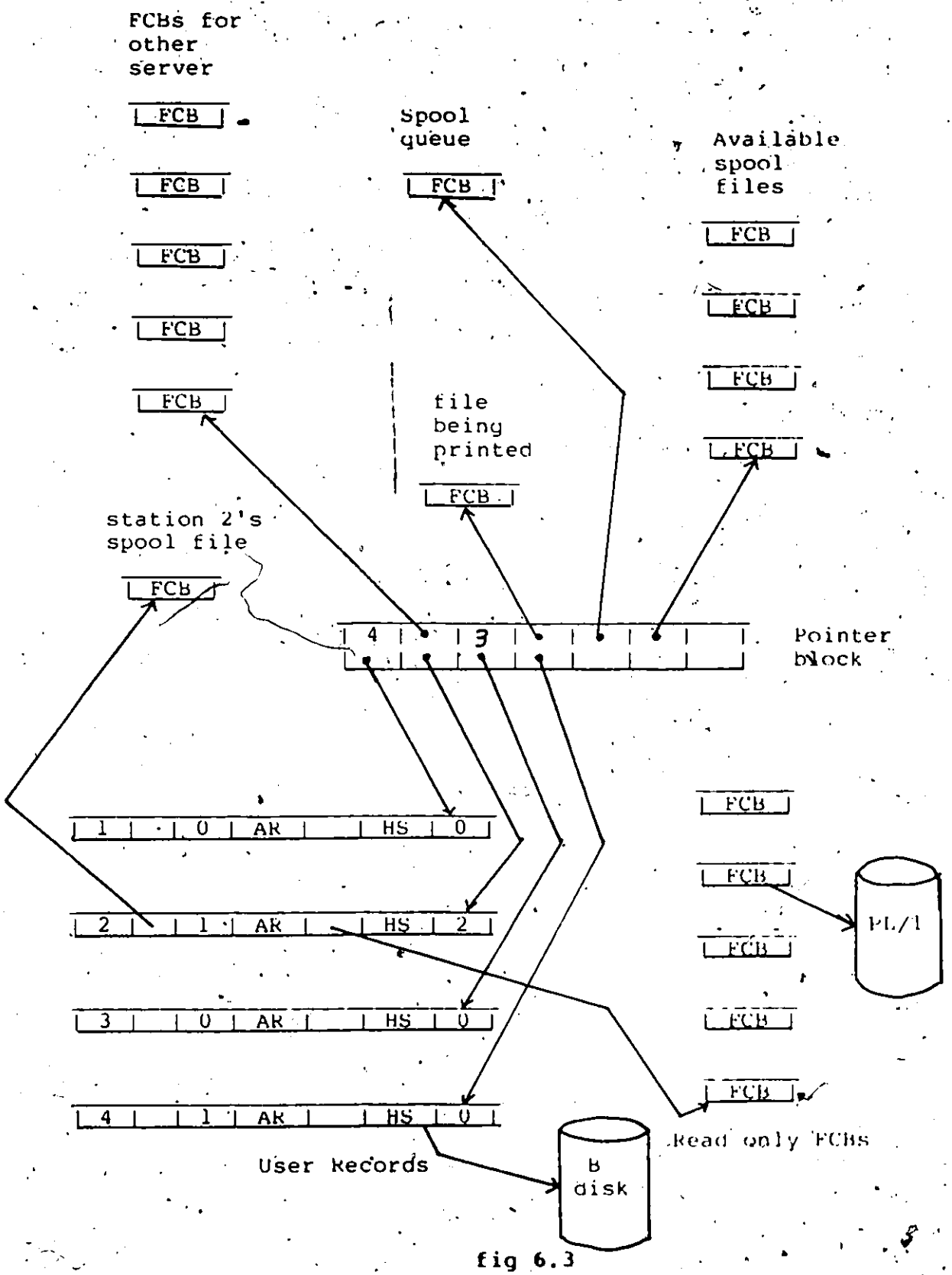


fig 6.3

System structures

being polled, there is one file being printed and an additional file in the print queue. Workstation 2 is currently spooling, but has not yet closed the spool file. Only workstations 2 and 4 have logged on.

6.4 Workstation Software. On the workstation everything is located in the BIOS. The BIOS of CPN is a superset of the standard CP/M BIOS. The jump vector at the beginning is the same, but there are some added functions following this, and there is a parameter area at a constant offset from the beginning of the BIOS.

6.4.1 CPN BIOS. The reader may wish to return to section 4.2.3 to review the standard BIOS jump table, as here we will only give the additions. They are:

```
        JMP   SPOOL
        JMP   MSG
        JMP   RED
        JMP   RESERVED
        JMP   RESERVED
PARM:   DS    ...           ; Parameter area.
```

The MSG routine takes as parameters pointers to a message and a buffer for the result. It will transmit the request to the file-server and return the result. It is essentially a call to the transport layer of the communication subsystem and is not currently used by any application programs. The SPOOL routine takes a parameter indicating the spool function to be performed. This

routine will be discussed in greater detail later. The RED routine is used to redirect the serial devices, such as redirecting the LST: to the screen. It is necessary for printing the page formatted output of COBOL programs. The reserved routine does nothing. It is reserved for future use.

The parameter area is initialized by the logon routine with information returned by the fileserver. It contains the user's name, the file name of the virtual disk, and its size. There are several reserved fields here as well.

6.4.2 Disk Operations. The routines, SELDSK, SETTRK, SETSEC, and SETDMA, do nothing but store the current disk, current track and sector, and the address where the data is to be transferred to or from. When the read or write routine is then called, the (track,sector) pair is converted to a single relative record which is used for getting the data.

When the request is for disk A: (the RAM disk), the relative record is used to find the page, and offset into the page of the banked memory. The data is then transferred to (from) the page from (to) the DMA address. On read requests, parity is checked to see if there is an error. If so, it indicates a hardware malfunction and a message indicating where it occurred is written on the screen before the workstation halts.

A request for the B: or C: disk is a request for a network disk, and must be sent to the fileserver. The first time an access is attempted, an OPEN message is sent. The index returned by the

fileserver is saved. A read (write) message is then sent. This message contains the index returned by the open, the relative record number and the data (in the case of a write). The message returned is the return code and the data (in the case of a read). If an error is returned by the fileserver, an open message is sent, and the operation is retried.

If a write is attempted on the C: disk, the BIOS will immediately return an error to the BDOS. This can be bypassed by setting a flag in the parameter area to zero. Knowledge of this, as well as having insert command give no special privileges to a user since security checking is done at the fileserver.

The INSERT virtual disk command allows a privileged user to "insert" any virtual disk into his(her) B: or C: drive. It works by moving the name of the virtual disk to be inserted into the file name area (in the BIOS parameter area) for the appropriate virtual disk drive, and then performing a cold boot. This will force an open before the next disk access.

6.4.3 Spooling - There is a flag in the BIOS called the spool flag. This flag is used to indicate if spooling is being done, and if so, if it is the console or LST: device. The CONOUT and LIST routines of the BIOS check the appropriate bits before writing the byte. If they are being spooled, the byte is sent to the spool routine as well.

The spool routine has several functions which are selected by a function number in a register. In the case of start spooling,

the bit in the spool flag is set to indicate which device is being spooled, and in the case of the console, an escape sequence is sent to the screen to set it to inverse video mode. A start spooling message is then sent to the fileserver. If an error code is returned, this indicates that the lab will be closing soon, and no more spool files are being accepted.

The send byte routine moves the spooled byte into a buffer. If the buffer is full, it is moved into a packet and sent to the fileserver, afterwards an empty buffer is initialized. The stop spool routine sends the partially full buffer and a close spool message to the fileserver. The screen is then reset to normal video.

The spool status routine has, as a parameter, the address of a buffer in which to receive a response. The routine sends a spool status request to the fileserver and receives the response in the buffer.

These spool routines are sometimes called by BIOS routines, and sometimes called by application programs. The SPOOL command calls all of them.

Chapter VII

Network Details

We will now look at how requests are passed from workstation to fileserver. As had been mentioned earlier, the network access method is polling. The fileserver polls each workstation to see if it has any requests. A poll which is not responded to, indicates to the fileserver that the workstation has no requests.

We will also examine communication between fileservers. This is done using a Corvus Omninet card. This card handles most of the low levels of communication without bothering the fileserver.

7.1 A Polled Network. Polling is the simplest scheme for communicating with workstations. It requires less hardware than other methods. For example, CSMA/CD requires the ability to "listen" to the wire while transmitting. Lets examine how polling works for us.

7.1.1 Message Interchange. The workstation will always transmit a request to the fileserver and expect a response. The fileserver will never send a message (other than the poll) to a workstation unless the workstation requests it. Although the fileserver is

the master of the network, it is a slave to the workstation.

The poll is responded to by an accept poll message rather than the request itself. This is done so that the timeout for the poll can be made small. A large message would require a longer timeout to ensure that the long messages would not be missed.

After sending the accept poll message, the workstation delays about 400 microseconds, then transmits its request. Much of the delay is used in setting up for the transmission, the remainder is to make ensure that the fileserver has enough time to set up itself for the receive. The workstation then waits for the acknowledgment from the fileserver. There is another poll, and the workstation must accept this poll to indicate that it is ready to receive the response. If the response has been received successfully, an acknowledgment is sent back to the fileserver. The process of sending a message and receiving the result takes 4 milliseconds. This does not include the time required to service the request.

Notice that there is a sequence for sending a message to the fileserver, and one for receiving the response. By the rules of this protocol, the sending and receiving of a message are independent. This may be useful at a later date. We have the possibility of getting requests from several stations before having to send back the response. This would improve performance if the operating system at the fileserver is changed to one which will allow I/O overlap on disk operations, and would sort the disk requests to minimize seek time.

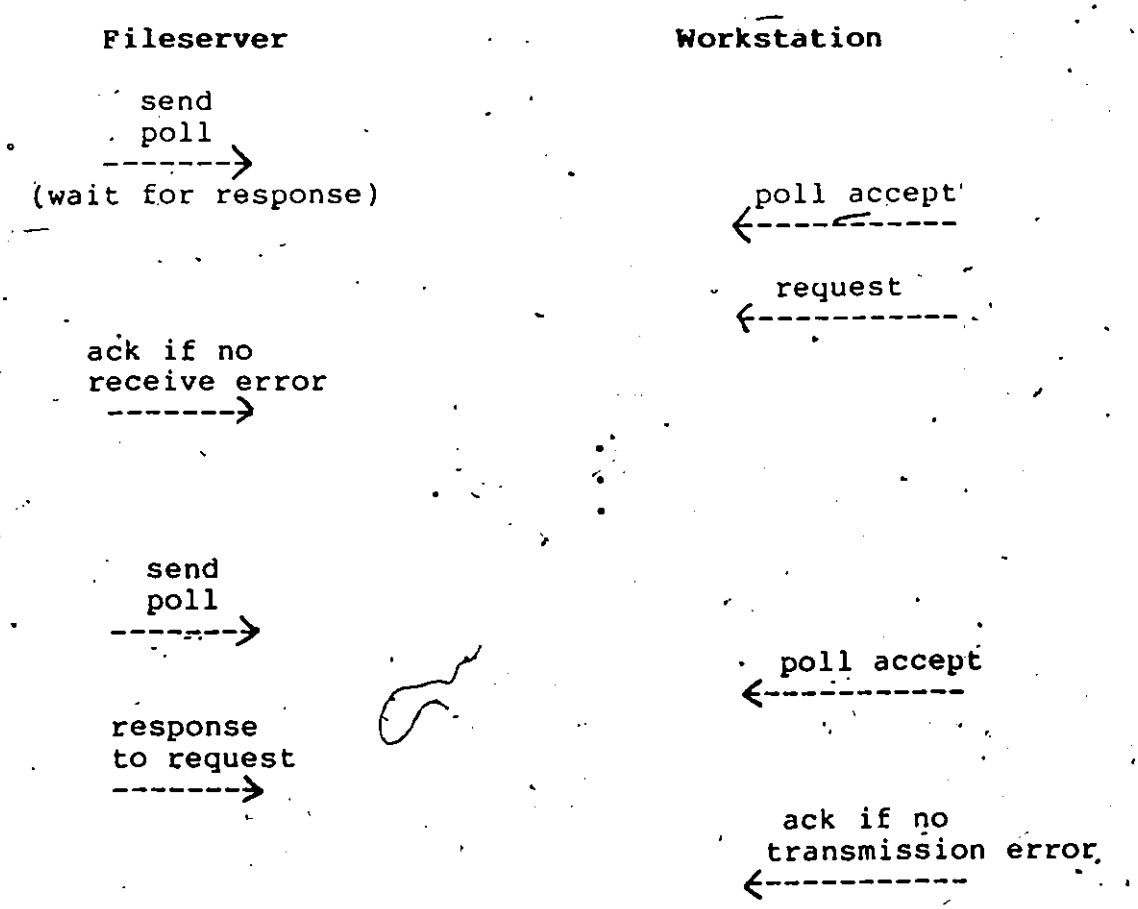


fig. 7.1
Message interchange.

Lets now look at what happens when there are errors or problems in the transmission. There are three things that can happen when a station expects to receive a message:

- nothing arrives,
- a garbled message arrives,
- something unexpected arrives.

If nothing arrives when something is expected, the communication routine will return a timeout error. Timeouts at the workstation

are very long, about 0.5 seconds. On the fileserver, they are much shorter since the workstation does not slow down on a busy network. If no message has arrived and a timeout error occurs, it is assumed that the sending station has crashed.

The assumption is not true if the station sent the message and either the address field or the leading flag were corrupted in transmission. In those cases, a sent message will never arrive. The probability of this happening is small, and its effect is not disastrous. The student would be informed of a communication error and would have to reboot his/her workstation. If it is the fileserver that does not receive the message, servicing of the request is aborted. This will cause a timeout or synchronization error at the workstation. If we were to try to recover from this type error, synchronization would be a problem in this polled environment.

When a garbled message arrives there will either be a CRC error or a data overrun error. Data overrun means that a message longer than the buffer is being received. Since the buffer is larger than any message that is ever sent, this is a serious problem. It means that an other station on the network has a hardware malfunction and is sending garbage continuously over the line. When this situation is detected, all communication halts, warning messages are printed. The problem causing workstation must be isolated before any communication can continue.

If there is a CRC error on a transmission, it could be control (poll, ack, etc.) or information. If a workstation receives a

garbled poll request, it is ignored, and it simply waits for the next poll. If the fileserver receives a garbled response from a poll, it assumes that it is the response expected. This poses no problem because if there is a problem, it will be detected further in the message exchanges. The workstation will only send one type of message in response to a poll, and only one station should be responding.

If a packet is received with a CRC error, a NAK packet is returned and there are up to 10 retries. The final problem comes when an expected ACK is garbled. This is handled simply by assuming that it is an ACK. This may sound like a problem, but it is not. If a NAK had been actually sent, the sender will be expecting a retransmission, but the other station will not do it, and the error will be detected then.

Finally, we come to receiving something unexpected. It means that a workstation is sending something when it shouldn't or its network address is the same as that of another station. This is a serious problem. It means that the synchronization is gone. Since each user has exclusive control of his/her machine, there is the possibility that: (1) the user could either perform a network I/O operation himself and not respect the line protocol or, (2) due to an uninitialized variable or array out of bounds, the message transmission routines are partially overwritten. When this happens, the offending workstations (There are usually two affected.) both die with an error message indicating the problem. It is better to halt processing when there is the chance of an error getting by undetected.

Appendix A gives full details with the state transition table for the message interchange.

The message interchange was written to help in locating hardware malfunctions. The transmission medium is quite effective and unrecoverable errors are infrequent. In the worst case, these errors (assuming that are not a result of hardware malfunctions) only require the user to retry the operation with a maximum loss of 2.5 minutes. This maximum would occur when a student is sending a 90K byte COBOL listing to be printed on an extremely busy day; and the error occurs on the last record transmitted.

7.1.2 Packet Format. The Zilog SIO chip is used for doing the communication. The messages are in HDLC format. This format is:

flag	information	CRC	flag

The flags and CRC are generated by the SIO chip. The information field must be supplied to the chip by the software. This format was chosen to be compatible with the message format of CP/Net. We do not loose anything by choosing to follow this format. It is:

DAD	FMT	SAD	FNC	SIZ	MSG

DAD - Destination address.
FMT - Message format.
SAD - Source address.
FNC - Function code.
SIZ - length of the MSG field.
MSG - The information for request/response.

The first byte is expected by the receiving hardware to be the destination address. The FMT field is used to indicate if the message is a request or response, as well as indicating if it is greater than 256 bytes long. The FNC field indicates which function (read, write, open, etc.) is to be performed. The contents of the MSG field are dependent on the function. Appendix B contains a list of the functions.

7.1.3 Analysis of the Polling. A problem with polling is that time is wasted polling the stations that do not respond. Lets see what the cost of that is to us. The time required to poll a workstation that does not respond is about 600 microseconds.

The worst case to examine the time spent on non-productive polls would be a single user in the second year lab. There would be one station requiring the fileserver, and 18 stations not needing it. Now to compute how much time is wasted for each poll of the entire network.

18 * 0.6 = 10.2 miliseconds

Lets take the case of that single user loading the ADA language system which is 256K bytes. Since the load is done in 2K blocks, there would be 128 message interchanges required to load it. There would then be

128 * 10.2 miliseconds = 1.3 seconds

of time wasted on non-productive polls in the worst case.

Notice that the worst cast for wasted time is when the network is empty. The more active stations there are, the less time there

is wasted. Polling seems to be a pretty good choice for us. If we were to use another scheme such as CSMA/CD, the busier the network would be, the greater the probability of collisions and the more time wasted. We waste time when we can afford to, and we don't when we can't.

7.2 Inter-fileserver Communication. The fileservers are equipped with a Corvus Omninet Transporter card. It provides for a fast (1M baud) and reliable communication with a minimal of CPU intervention. The board consists of a processor, RAM, Serial I/O, and a DMA.

In order to transmit a message, the message and a control block must be transferred into the Omninet board's memory. A command is then given to the transporter's processor. The main CPU is then free to do anything else while the processor on the transporter board will transmit the message to the desired station and wait for an acknowledgment. If there is no acknowledgment, the message will be retransmitted up to 127 times. After completion of transmission, either successful or not, a condition code is set indicating the status of the transmission. The main CPU is not aware that this is taking place, and, since the memory is not associated with it, no memory cycles are stolen from it.

In order to receive a message, a "socket" for the message is set up by passing a control block to the transporter board. Up to four sockets can be set up and each one is capable of receiving a message. Once set up, the transporter will receive and acknowl-

edge messages, again without intervention. One socket is used for receiving the response to a request, while the other three are used when other fileservers have a request for this one.

The line interface is an RS-422 driver on a single pair of wires. CSMA/CD is used for line access, and the transmission follows the ADLC protocol.

This board was chosen for inter-fileserver communication due to the fact that once set up, it can receive up to 4 messages with no CPU intervention. It is easy to prevent a deadlock situation between the file servers. In addition, it is easy to justify polling on the workstation network because there is the file-server that is the master of the network, but what would be considered master of a network of five fileservers? The corvus let each fileserver be equal. We can bring down any one and there will be no effect (to users not wanting that fileserver) on the network.equipped with a Corvus Omninet Transporter card.

Chapter VII

Concluding Remarks

We now see that the CPN has given us a lab that is modern, reliable, efficient to run, and versatile. Problems associated with other lab organizations have been eliminated, giving the students a system that is a pleasure to use.

The high cost of buying and maintaining a mainframe or minicomputer has been eliminated. Microcomputers can be maintained by the department's technicians giving a more reliable system. Numerous problems associated with microcomputers have been eliminated. The diskless workstations offer

- no drives to break down,
- no problems distributing software,
- and no problem with theft of software.

Problems of software differences and incompatabilities have been eliminated by careful choosing of application software. CPN is a single integrated system incorporating software products from different vendors.

A philosophy used in CPN is: Keep things simple. Changes which could have been made to improve the speed were not done in order

keep things simple. For example, the fileserver could try to optimize disk seeks by polling all stations, computing the addresses of the sectors to be read or written, and then issue the operation. This is essentially bypassing the operating system on which the fileserver is running. On any system a speed up can be achieved, if the operating system is not used because the program running knows its needs better. Needless to say, this is rarely done. The ends do not always justify the means.

In the 18 months of its use, the system has been well liked by students. The reasons that they like it is that it is easy to use, good response time, and there is good documentation available for them.

As far as hardware maintenance goes, a single technician spends less than one day a week on their maintenance. There has not been any software maintenance (improvements, bug fixes, etc.) performed in the past year other than account creation and deletion.

The lab has lived up to our expectations. Originally we had planned that the equipment would have life of 5 years before becoming obsolete. Since the lab is running so well, that may be extended. The only problem is the growing popularity of the IBM PC, and the fact that most new software is aimed at that one machine.

8.1 Problems with CPN. There are several areas that in theory are a problem, but in practice they are not. They are in areas of network security. Any network of microcomputers has similar problems. When a user has exclusive control of a machine, as well as programming tools, and that machine is the one that sends messages, there is no way of preventing him/her from issuing I/O to/from the network. A malicious user could:

- receive messages intended for the fileserver,
- continuously send messages, thereby load the network,
- sending junk onto the line to destroy all communications.

Due to the hardware used with CPN, all three are possible. There have been no problems and none are expected because:

- Students don't have time to fool around and the SIO chip is very complicated to program.
- There is no "reward" for crashing the system. Only the other students would suffer.
- There is nothing of value to be gained by "breaking into" the system.

8.2 Further work. The next phase of CPN is to make it include a complete lab monitor as well as machines to work on. There are always peak times, just before assignments are due, when there are not enough workstations for the students. CPN should be able to do something about this.

The lab has two kinds of usage: scheduled labs in which there is a teaching assistant in the lab, and free time. During the free time, anyone can use a workstation. During peak times, students

may have to wait quite a while until there is a free station.

During the scheduled, the lab should contain only students taking a particular course. During the peak periods, other students sneak in to work on something else, taking the workstation of a student that should be in the lab. At these times, the assistant may spend more time kicking people out than helping his/her students.

A reservation system could be set up that would solve the two problems. The account record already contains the list of courses and lab sections that the student is enrolled in. It would be a simple matter to have the fileserver look at a time table for the labs, and if it is the time of a scheduled lab, only allow students registered for that lab to log on. Since the workstations are practically stand alone microcomputers once a logon has been done, students that were in the lab already would not be effected by this. Modifications could be done to the BIOS of the workstation that would cause a simple query at each warm boot. If a message is returned saying that it is time to go, the student would get a finish up quick message and eventually be logged out. The only additional hardware requirements would be a real time clock at the fileserver. Software modifications would be straightforward.

The problem of monitoring the free time is a little more involved. Students could be given a certain number of hours per week of lab use. The amount of time would depend on which courses (s)he is taking. A special station could be set up out-

side the lab that would let a student reserve time slots throughout the week. This station could send the schedules to the fileserver. The fileservers would then use these schedules similar to the way it is done for scheduled labs. This would have the advantage for students that they would know in advance that there is a workstation available.

Another area of work would be to add other microcomputers to the network. This would probably involve building custom network interfaces for the machine, or an RS-232 gateway. These stations could be more popular machines (eg. IBM PC) and have disks. If an RS-232 gateway is built, the network could be accessed (although slowly) via telephone lines. This, unfortunately, would not be available to the students for at home work. A network that could distribute copyrighted software over a city would have to be kept secret, and it is not certain that 1000 students could keep the number secret.

The RS-232 gateway would enable the lab to be extended to spread over several buildings within the university using higher speed modems. The access via this gateway would be slower than direct access, but it could be used by a professor to look at a student's problem without leaving his office.

APPENDIX A

State Transitions for Message Interchange

Workstation

STATE	ACTION	Next State
1 Wait for poll		
	Receive poll OK	2
	Timeout	11
	Receive message,not poll	12
	CRC error	1
2 Send ACK		
	Transmit OK	3
	Unable to transmit	13
3 Send request		
	Transmit OK	4
	Unable to transmit	13
4 Wait for ACK		
	Receive ACK	5
	Receive NAK	3
	Timeout	11
	Receive other message	12
	CRC error	5
5 Wait for poll		
	Receive poll OK	6
	Timeout	11
	Receive message/not poll	12
	CRC error	14
6 Send ACK		
	Transmit OK	7
	Unable to transmit	13
7 Wait for response		
	Receive OK	8
	Timeout	11
	Receive other message	12
	CRC error	9
8 Send ACK		
	Transmit OK	10
	Unable to transmit	13
9 Send NAK		
	Transmit OK	7
	Unable to transmit	13

10 Interchange complete.

11 Fileserver is down.

12 Two nodes with same address.

13 Hardware malfunction

14 Temporary problems, abort interchange.

State Transitions for Message Interchange

Fileserver

STATE	ACTION	Next State
1 Send poll		
	Transmit OK	2
	Unable to transmit	14
2 Wait for ACK		
	Receive OK	3
	Timeout	12
	Receive other message	13
	CRC error	13
3 Wait for request		
	Receive OK	4
	Timeout	13
	Receive other message	13
	CRC error	5
4 Send Ack		
	Transmit OK	6
	Unable to transmit	14
5 Send NAK		
	Transmit OK	2
	Unable to transmit	14
6 service the request		7
7 Send poll.		
	Transmit OK	8
	Unable to transmit	14
8 Wait for ACK		
	Receive OK	9
	Timeout	12
	Receive other message	13
	CRC error	13
9 Send response		
	Transmit OK	10
	Unable to transmit	14
10 Wait ACK		
	Receive OK	11
	Timeout	12
	Receive other message	13
	CRC error	11

11 Interchange completed successfully.

12 Station has no request.

13 Interchange aborted.

14 Hardware malfunction.

APPENDIX B

Message Formats

OPEN request:

| DAD | 00 | SAD | 0F | 0D | DRV | FN | FT | EXT |

OPEN response:

| DAD | 01 | SAD | 0F | 03 | IDX | FS | RC |

DAD - server's NID.
SAD - Workstation's NID.
DRV - Disk drive.
FN - File name.
FT - File type.
EXT - Extent number.
IDX - Index for server's list.
FS - File size.
RC - Return code.

READ request:

| DAD | 00 | SAD | 21 | 05 | IDX | RCN |

READ response:

| DAD | 01 | SAD | 21 | 83 | data | RC |

DAD - server's NID.
SAD - Workstation's NID.
IDX - Index for server's list.
RCN - Record number to be read.
RC - Return code.

RITE request:

| DAD | 00 | SAD | 22 | 85 | IDX | data | RCN |

WRITE response

| DAD | 01 | SAD | 22 | 01 | RC |

DAD - server's NID.
SAD - Workstation's NID.
IDX - Index for server's list.
RCN - Record number to be read.
RC - Return code.

LOAD request:

| DAD | 02 | SAD | 41 | 0405 | IDX | data | RCN |

LOAD response

| DAD | 01 | SAD | 21 | 01 | RC |

DAD - server's NID.
SAD - Workstation's NID.
IDX - Index for server's list.
RCN - Record number to be read.
RC - Return Code.

SPOOL request:

DAD	00	SAD	05	nn	SSF	data

SPOOL response:

DAD	01	SAD	05	nn	SSF	data

DAD - server's NID.
SAD - Workstation's NID.
nn - Length-1 of data field.
SSF - Spool subfunction number.

references

Ahuja, Vijay, "Design and Analysis of Computer Communication Networks", McGraw-Hill, Inc., 1982

Bennet, J.M. and Chefurka, P.V., "Micronet: A Low-cost Local Area Network for Microcomputers", Proc. 1983 ACM Conference on Personal and Small Computers, Vol. 6, No. 2, Dec. 1983, pp 209-216

Chorafas, Dimitris N., "Designing and Implementing Local Area Networks", McGraw-Hill, Inc., 1982

Christopher, T. W, Hatcher, P.J., "A Network Computer for Distributed Software Research", Proc. 1983 ACM Conference on Personal and Small Computers, Vol. 6, No. 2, Dec. 1983, pp 9-13

Clapp, G. H, "An Analysis of CP/NET", Proc. 1983 ACM Conference on Personal and Small Computers, Vol. 6, No. 2, Dec. 1983, pp 117-124

Dahmke, Mark, "Microcomputer Operating Systems", Byte Books, 1982

Digital Research, Inc., "CP/NET Operating System Reference Manual", Digital Research, Inc., Pacific Grove, CA., 5th Edition, 1982

Digital Research, Inc., "CP/M Plus Operating system System Guide", Digital Research, Inc., Pacific Grove, CA., 1st Edition, 1982

Digital Research, Inc., "CP/M 2.2 Alteration Guide", Digital Research, Inc., Pacific Grove, CA.

Digital Research, Inc., "CP/M 2.2 Interface guide", Digital Research, Inc., Pacific Grove, CA.

Dy-4 Systems, Inc., "Operations Manual DSTD-102, DSTD-325,DSTD-401", DY-4 Systems, Inc. Ottawa, Ont., 1982

Gee, K C E, "An Introduction to Open Systems Interconnection.", NCC Publications, 1980

Hickey, P.J. Raymond, J., "An Operating Environment for Educational Microcomputer Networks", Proc. 25th Symposium on Microcomputers and their Applications, June 1984, pp 1-5

Hickey, Peter J., "CPN Users Manual", CSIUO Publishing, 1984

Hogan, Thomas, "CP/M User Guide", Osborne.Mcgraw-Hill, Berkeley, CA., 1982

Leventhal, Lance A., "Z80 Assembly Language Programming", Osborne/McGraw-Hill, 1979

Peterson, J Silberschatz A, "Operating System Concepts", Addison-Wesley Publishing Compa, 1983

Rolander, Thomas A. et al, "Network Software borrows design from microcomputer operating system", Data Communications, Vol. 11, No. 13, pp 123-133

Satou, K. et al, "Design and Implementation of a Personal Computer Local Network", Proc. 1983 ACM Conference on Personal and Small Computers, Vol. 6, No. 2, Dec. 1983, pp 222-231

Shaw, Alan C., "The Logical Design of Operating Systems", Prentice-Hall, Inc. Englewood,, 1974

Sherman, Kenneth, "Data Communications: a users guide", Reston Publishing Company, Inc., 1981

Softech Microsystems, Inc., "UCSD p-System Installation guide", Softech Microsystems, Inc., 1st Edition, Feb. 1982

Solnsteff, "A distributed OS for an Educational Microcomputer Net", Proc. Third Symposium on smalls, vol. 6, No 2, Sept. 1980, pp 67-71

Stritter, E. P., Saal, H., Shustek, L, "Local Networks of Personal Computers", Proc. COMPSCON Spring 81, Feb. 1981, pp 2-5

Summers, R.C. et al, "RM: A Resource-Sharing System for Personal Computers", Proc. 1983 ACM Conference on Personal and Small Computers, Vol. 6, No. 2, Dec. 1983, pp 91-97

Tanembaum, Andrew S, "Network Protocols", Computing Surveys, Vol 13, No. 4, Dec. 1981, pp 454-489

Tanenbaum, Andrew S., "Computer Networks", Prentice-Hall, Inc., Englewood Cliffs, N.J., 1981

Ward, Sandra, "Personal Workstations at the University of Waterloo", Proc. ACM-SIGUCCS, Vol. 12, Nov. 1984, pp 37-43

Zilog, Inc., "Using the Z80 SIO with SDLC" (document #617-1564-0007),Zilog, Inc., 1980

Zilog, Inc., "A Z80 based System Using the DMA with the SIO", (document #715-1809-0002),Zilog, Inc., 1981

Abstract

We describe the requirements, design, and functions for a Computer Science laboratory at the University level. The laboratory consists of networks of microcomputers and it services the first two years of a typical computer science curriculum. This includes both courses in practical programming (Pascal, ADA, Prolog, etc.) as well as support for more theoretically oriented courses (Problem solving, data structures, grammar concepts).

The software environment includes:

- Compilers, interpreters, word processing, spreadsheets, and database packages from outside vendors.
- An operating system compatible with CP/M compatible software
- A file server controller and spooling facilities.
- A network controller system.

All of these components have been integrated into one system, fine tuned to the requirements of a student environment and university needs. These components include:

- Smart student friendliness
- Incapability of copying systems files, compilers, and the files of other students
- Broadcasting functions for assignments distribution and correction, for mark processing and soft copy correction
- Low operational cost, low software maintenance
- Ease of change with curriculum, or software market products
- High access to students, quick turn around.

This system provides significant benefits over existing local area networks of microcomputers both in terms of functionality and in terms of performance.