SHARE

SHARE SESSION REPORT

61	C209	Series/1 DDp at Watchtower	24
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE
Local Area Networks		Barry Lay	TY
PROJECT		SESSION CHAIRMAN	INST. CODE

University of Toronto Computing Services, 255 Huron St., Toronto, Ontario, Canada, (416) 978-3328

SESSION CHAIRMAN'S COMPANY, ADDRESS, AND PHONE NUMBER

ABSTRACT

Use of the S/1 as a front end text capture tool offers high function at low cost. The problems of communications, word processing, and file management in a distributed environment will be discussed. The high speed 370 channel interface on the S/1 is used as a communications tool for host data base file and retrieval capabilities. Use of the 3101 as an interactive user friendly word processing terminal on the Series/1 will be presented and contracted with use of the Personal Computer.

Distributed Processing is usually an important part of Data Processing. We at Watchtower have found it necessary in our installation to make use of this valuable function. It is the evaluation of distributed editors and functions in a 370 host environment that is the subject of this presentation.

The Series/1 as a front end text capture tool offers the capability of off-loading many functions from the host system. On the Series/1 resides an editor. This editor is able to access local Series/1 disk as well as supporting many terminal users of that editor. The Series/1 also has a communication link to the host System/370 running VM. It is able to communicate with a data base residing there. The advantages of using a distributed editor on the Series/1 are (1) editors that normally run on the 3270 type terminals would now no longer require host cycles to do their work. (2) in the event that the host system would be down users of the Series/1 editor could continue to work locally employing the Series/1 disk.

In working towards the implementation of the Series/1 distributed editor, there were many different tasks that had to be accomplished. The first of these to be discussed is communications. In order for any intelligent device to be useful to a host system it is necessary that this device communicate with it. One of our problems in writing a host distributed Series/1 editor was that of file communications. Secondly, in addition to communicating files or buffers between the two processors, it is important to consider the security of the host data base and how the access mechanism of data in the host would work. Thirdly, there was the question of word processing. What type of word processing would be used on the Series/1? Currently there are several editors available for the Series/1, yet none of them seemed to fit our needs or seemed to integrate into the file communications structure that we needed.

Therefore, it was decided that we would write our own editor for word processing. We chose to do our initial development using the IBM 3101, a TTY-type terminal of low cost. We felt that if we could make this low cost terminal perform in a Series/1 environment then it would make the best use of our available resources, since numerous 3101s had already been purchased. After spending many months of research, programming, testing and prototyping, it is appropriate to present our findings, evaluations and conclusions about the Series/1, its use, the 3101 editor and other distributed data processing possiblities.

The first topic of communications is the environment. Where will my distributed processor be located? Which options are available in my host? What cabling is already installed or which is the most desirable to install if none are currently available? What amount of through-put will there be on this link? Will the users be interactive or batch? The answers to these questions will aid the user in clearly seeing which solution best fits his requirements.

Our Series/1s were within the same Computer Room as our 370 System. Therefore, we chose a channel attachment interface from the Series/1 to the host. On the Series/1 it was necessary to write a device driver for the channel attachment hardware. On the VM/370 host it was necessary to write within a CMS machine a device controller and interrupt handler. The

Series/1 appeared to the host to be managing many real addresses, these real addresses being attached as virtual addresses to the user written CMS controller.

What did the Series/1 channel hardware and device driver do? The Series/1 device driver and hardware emulated a 32-device 3270 controller. What type of controller? A 3272? Or a 3274? Actually the answer is none at all. The console I/O support of VM CP was not used for this interface. The Series/1 to host interface was a strictly defined Start I/O protocol which supports READ, WRITE and ATTENTION only.

What happens within the Series/1 when the host sends an interrupt? The Series/1 hardware transfers control to our interrupt handler. It checks the device control block which is set up by our application program to find the status of that hardware address. If the device status is valid the I/O drivers use the device control block to perform the I/O requested by the host in behalf of the appropriate Series/1 application. When the I/O is complete the Series/1 application program with which the host is communicating is informed. It can now continue as required.

What happens on the host when the Series/1 generates an interrupt? CP turns control of the interrupt over to the interrupt handler in our CMS application program. It interrogates the current CSW, channel status word, and determines the address of the interrupting device as well as its interrupt code. Using this information, the application control block is searched to determine if this interrupt was appropriate. Appropriate status information and address information regarding any data are saved in the control block. The interrupt handler posts the CMS device controller application informing it that there is work to be done. The device controller searches the control blocks and processes any outstanding work. This may include the building of CCWs and the execution of Start I/Os. When all work is done the controller waits on the interrupt handler to signal it to begin again.

It is interesting to note that by the use of an interrupt handler in conjunction with an application program and by the careful use of control blocks in the form of status tables that a multi-tasking environment within CMS can be simulated.

We have talked about how interrupts are handled in both the host and Series/1. How, then, is a block of text sent from one processor to another. The sending of a block of data is driven by the Series/1. This is comparable to the way a 3277 terminal operator drives an application program. When that operator hits ENTER, it signals the host application to read data. The host application processes that data and writes the appropriate reply back to the 3277. The Series/1, like the 3277, always does a WRITE to the host followed by a READ from the host. To send a block of text the Series/1: 1) generates an attention to the host, the host responds by doing a READ, requesting that the data on the Series/1 be sent to it, 2) the Series/1 writes its data to the host, the host processes that data and finishes the block transfer by writing its response to the Series/1. Data transfers are fully interlocked. If the Series/1 were sending a file of many blocks to the host, it could not write them to the host one after another. The host must acknowledge each Series/1 block by a WRITE. The protocol expects every READ to be followed by a corresponding WRITE.

The next area of concern is data security and access management. What are some of the objectives of a security and access management system? What were our objectives? One is that the host must protect its data base. It was felt inappropriate to place the responsibility for protection in the Series/1, a distributed user. Therefore all requests made by the Series/1 for data transfers need to be validated by a security mechanism within the host itself. It was determined that this mechanism should be as close to the file access mechanism as possible. In our description so far a CMS interrupt handler and a device controller were mentioned. These also should not validate requests from the Series/1 for files as they are too far removed from the actual data. Those programs that directly access the data base are in the best position to evaluate any user requests and are thus able to best protect the system's integrity.

Once a block of data or a request for data has been received by the CMS device controller how is that file access accomplished? Our file access mechanism has three parts, the device controller, the security server and the file server, which directly connects to our data base. The device controller sends a block or a request which it has received from the Series/1 to the security server by means of VMCF. That device is marked as waiting for a reply. The VMCF message from the controller will cause an INTERRUPT in the interrupt handler of the security server who saves the VMCF request block. The security server application is posted that work is to be done. The request block or data block contains security and function information. This allows the security server to validate the work it receives. Besides request validation, data may need to be formatted or buffered to make it usable by the file server. This is done in the security server. VMCF is used to send to the file server validated blocks or requests. The security server marks the request as outstanding and continues with any other work it has to do.

The file server is interrupted and eventually receives and processes the security server request. The data is read from or written to the data base according to the control information supplied by the security server. When this is complete, a reply by means of VMCF is made to the security server. The

security server again receives an external INTERRUPT, this time from the file server. This reply from the file server is analyzed to determine what was done and where the reply should go. The reply is forwarded by means of VMCF to the device controller. The device controller is interrupted by the VMCF response. When the response is processed an indication that the operation is complete is sent from the device controller back to the Series/1.

In addition to simply being able to access the information in our host data base, it is necessary to keep those files secure. How is this done? Our data base is divided into two sections-- a directory and a data library. There is a third portion called Groups and Users. Groups and Users is actually a file that is used in conjunction with the directory to provide data protection. The directory contains the publication and unit names of all files in our data base. In addition to that, such things as the authorization for READ/WRITE privileges, READ ONLY privileges, the date of creation, the date of last update, the publication status and block key information are found in the directory.

The data library consists of logically linked blocks of data. Each file is given a unique block key which identifies it. Within the file there is also a relative key which maintains the logical order of the file.

The directory which contains the authorization is checked against the group and user file when a request is made to access a data file. If the requesting user is a member of the set of users with access privileges to the file then access is given for that request. Many groups or users may have authorization to a file at a time, but only one may update it at any time. Flexibility is provided by the use of groups which may contain many users or even other groups. If a group were authorized to READ & WRITE a file then any user in that group or any user in any group assigned to the authorized group could gain access to that file. This allows users working together to easily share data while controlling use of the data base as a whole.

Word processing was done on the Series/1. This meant that there would be a need for paging. Why is paging needed? Basically because the Series/1 is a 512K system. We intended to support 32 users per Series/1. This left little work space for each user in memory. In order to make efficient use of this limited storage it was evident that a file paging system be used.

With any paging system there are several questions that need to be answered regarding page storage and free space management. We chose to link pages of a file together using forward and backward pointers. When a user reads a file it becomes a working copy in his paging area. A directory entry for that user is made in the paging space for the copied file. This entry is a known location to the editor such that when the copy is referenced it need not be searched for but can be directly accessed. As work progressed on our paging system we also were writing a local file manager for Series/1 disk and a printer spooling routine. These tasks both required disk record management which was very similar to that used by paging. Therefore it was decided that instead of having three systems doing disk record management there would be one. Instead of dividing the disk into three separate pools of records we used but one big area shared by all. Each task (paging, spooling and local data base) would need its own directory, but because directories are treated like files this posed no real problem. This concept saved much work and some valuable main storage.

How does paging work? Paging is a black box, meaning that the internals of paging are not known to any one of the application systems that use it. The editor knows about only one line at a time, the current line. If the editor needs to move forward in the file it makes a request to paging for the address of the next line. By requesting previous and next lines from paging the editor can perform all its functions. Because of this concept only paging knows when and where to page the various pages. At sign-on each user is given a fixed number of pages which become his private pool. Paging manipulates the text of the file currently being edited by the user to and from these private pools.

When a file is read, that user has an exclusive copy of that file. No one else can access that data. Free space is treated as a file, and yet free space is different in that all users need to access it at any time. In order to prevent the free space file's chain of forward and backward pointers from being broken only one user at a time may use it. For the duration of the critical breaking and relinking of chains the requesting user enqueues the resource, quickly performs his operation and then releases the chain for others to use.

In addition to free space, the problem of how to protect chains from being broken with low overhead was considered. It is possible to specify a link order in which it is impossible for the chains to be inadvertently broken. Yet we found that this resulted in large amounts of overhead. By writing a RESTORE program we were able to cut overhead and improve response times. The RESTORE program, by tracing pointers in both directions, is able to recreate all files completely. The only exception is when the program is caught just at the point where it had broken a chain. That file would lose the temporarily free page, but the rest of the file would be intact.

An area of concern in our Series/1 word processing was screen management. This was made especially noticeable by our goal of using the 3101 terminal. Output processing has to take into consideration that at 9600 bps (bits pre second) it takes two seconds to repaint the 1920 character screen of the 3101. This means that when the user does real time interactive scrolling it could take two seconds to display the results if each line were individually scrolled. This user interface restriction is not acceptable. Scrolling is the worst case in terms of output processing because it requires the most data to be displayed in the shortest period of time. What could be done to correct or circumvent this problem?

What we determined was that by using leaps, small vertical windows and some other tools, many of the problems inherent in the 3101 could be circumvented. If the user was not to scroll one line at a time but do leaps of a quarter page, a half-page or any other user specifiable amount, this would mean that the scrolling or repainting of the screen could be limited in its occurrences. By dividing the screen vertically in half or in quarters, the amount of time that it would take to repaint the text in any given portion of that screen would also likewise be reduced. By presenting only a part of the window during consecutive scrolls the user can see enough of the result of his operation to make it useful and yet reduce the time required to do each display. When the last scroll in a consecutive scrolling operation has been processed the entire screen is painted showing the final results. By optimizing output using erase end of line and erase end of screen it is possible to improve the responsiveness of the presentation to the user.

Another output problem centers around the limited highlighting capabilities of a 3101 in Character Mode. How do you show defined blocks for MOVES, COPIES or DELETES when there is neither high or low intensity available in the hardware? We found that by temporarily translating all text to upper case for defined blocks the user is given an adequate indication of the area with which he is working.

One of the requirements of state-of-the-art word processors today is that of interactive character processing. 3270-type terminals are not interactive as they require that ENTER be hit in order to transmit a character or block of characters to the processor. Characters on the Series/1 are captured as they occur. These keystrokes are buffered into a user input area from which the editor retrieves them. The editor can react to each keystroke as it occurs displaying the results to the user.

All of the editor modules (REMOTE READ and WRITE, LOCAL READ and WRITE, PAGING, PRINT, SCREEN MANAGEMENT and the EDITOR itself) were designed as black boxes. Each was defined as an interface to the other. Thus if the internals of any module were changed, the interface and operation of the other modules was not affected.

As we neared the completion of our 3101 Series/1 distributed word processor, it became evident that other options were available. In comparing the 3101 against the IBM Personal Computer (PC) as an intelligent terminal, the restriction of a 9600 bps refresh rate and the limited highlighting options for the 3101 screen made it less desirable. The 3101 hardware character generator also limited the character set available for user display.

The PC, on the other hand, had fast access to the screen with memory mapped screen display. The many highlighting options of the PC terminal made its presentation of text to the user more friendly. Not only could defined blocks be clearly presented, but even such things as underline, blinking, reverse video, etc could be used to help make clear to the user exactly what he is doing. A PC with a graphics attachment and color display monitor could also be programmed to display soft characters in graphic mode. Given enough time font styles such as bold and italic could be shown on the PC.

What were the advantages of the 3101 terminal? At the time we began our project we had a relatively large number of these 3101 terminals. To write a distributed editor for them would make good use of currently existing resources. On the other hand, we at that time had no PCs. To buy PCs and write an editor for a distributed environment would mean a large investment.

The PC processor can also be compared to the Series/1. The Series/1 is very good at easily controlling many terminals and devices. This is due in part to its multi-tasking supervisor as well as its four-level hardware interrupt architecture. The Series/1 can access large amounts of disk storage. On the other hand the PC gives each user his own processor and main memory equal to that of a Series/1. On the PC this meant that paging for word processing could be simple. There is also no contention for shared resources within the PC processor since each user is self-sufficient. Because of this we are able to use high level languages such as PASCAL, C or PL/1 on the PC. On the Series/1, ASSEMBLER language was needed to write reentrant cross partition code. These considerations provide a three to one increase in programmer productivity on the PC.

In addition to channel attach, BISYNC, SDLC and ASYNC communications were used. PCs connected to Series/1 with 5250 twinax or Ethernet coax for high speed data transfer was also considered. It is hard to compare these different communication options to one another. Rather it is more important to know what can and can not be done with each. With this knowl-

edge the right communications can be selected to meet organizational needs.

BISYNC and SDLC are good for long distance communications. Using modems, communications to anywhere can be achieved. Additionally, the BISYNCH and SDLC cable is relatively small in size, meaning that it can easily be run within a building without much trouble. On the other hand, the remote communication rate of these two methods is slow, usually somewhere in the area of 9600 bps.

Twinax from the Series/1 using 5250 emulation on the PC is good in that it gives us a high-speed 500K bps interface. This cabling too is small, allowing it to be strung easily throughout our buildings. On the other hand, it is a local connect, meaning that the maximum distance from the Series/1 to a PC is less than 5,000 feet.

The ASYNC connection between the PC and the Series/1, the PC and the host and the 3101 and the Series/1 allows for long distance communications with the use of modems. Since the cabling is quad-wire, it is convenient to use. Often quad-wire is found already in buildings for its telephone system. Yet the 9600 bps communication rate makes interactive processing difficult if not impossible.

The 370 channel interface to the Series/1 allowed us a super-fast 1600k bps interface for high-speed data transfers between our host data base and a distributed Series/1 processor. If the distributed intelligent terminal were not local or if it were not capable of a host channel interface, such as is the case of a PC workstation, that device would need some other processor or interface to gain high speed access to a 370 host.

In summary, we found that the Series/1 is an excellent smart device controller and protocol converter. The PC on the other hand functions well as a distributed intelligent terminal connected to a controller such as the Series/1.

What, then, is our current strategy? We intend to use PCs as a multi-function workstation which can perform pass-through emulations of 3270, 5250 and 4978 terminals, as well as distributed host functions. The Series/1 can be used as a virtual PC diskette, network controller and a device controller for OEM devices.

As the data processing industry changes, user expectations and requirements change. In order to meet such needs data processing must be able to make use of those tools that become available.

SESSION REPORT

SHARE

61	C307	Managing Personal Computers in the Corp. Env.	600
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE

Integrated Personal Computer	C. Wrandle Barth	CSR
PROJECT	SESSION CHAIRMAN	INST. CODE

Computer Sciences Corp., 11700 Montgomery Rd., Beltsville, MD 20705
SESSION CHAIRMAN'S COMPANY, ADDRESS, and PHONE NUMBER

MANAGING PERSONAL COMPUTERS IN THE CORPORATE ENVIRONMENT

John Gosden and Joy Strasser

The Equitable Life Assurance Society of the United States
1285 Avenue of the Americas
New York, N.Y. 10019

Installation Code: ELA

Integrated Personal Computers Project

Session Number C307

ABSTRACT

As the use of Personal Computers continues to expand in the corporate environment, proper management of them involves maintaining a good balance between encouragement and control.

Personal Computer Growth

In only a few years, the use of personal computers in the office has grown very rapidly. The growth is fueled by a number of factors: general consumer expectation, computer-minded staff, decreasing cost and increasing capacity of personal computers, availability of good software and, of course, the nationwide advertising and enthusiastic articles extolling the wonders of personal computers for the office.

A major factor behind the growth of personal computers for both office and home is consumer expectation of faster and more useful services. The message from consumers is "I Want It Now". Marketers have responded with cash machines and extended business hours, 800 phone numbers and shop-at-home services. Computer services that have been marketed for the home computers include: electronic mail, stock quotations and airline flight information; while cable TV supplies the local weather, community news and first-run movies.

Employees bring their expectations of faster and easier to use service to the office. While most corporations have their major business systems computerized and many of them have word processing to support the general office, they lack systems to meet rapidly growing management information needs. When managers find they cannot get the information they need from the data processing department in the timeframe they need it, they are motivated to buy personal computer systems of their own to meet these needs.