Discussed is a computer network experiment designed to study a method of making the computing power of a high-speed batch system available to the interactive terminal system user. Commands for effecting the intersystem linkage and network operations are presented. Emphasized are system measurements and system tuning techniques for increasing efficiency.

Evaluation of an interactive-batch system network

by W. S. Hobgood

There are numerous large computer installations in the United States—predominantly in Government and university environments—that offer two distinct system services: (1) high-computing-power batch processing within the computing center; and (2) interactive, time-shared data processing in a terminal-oriented environment. Although many such locations offer side-by-side, interactive and batch systems, few have direct communication linkages between the two. Thus a fruitful area of programming research is the creation of effective linkages between these two services, especially where the research effort includes analysis of the manner in which the linkages are used.

network linkages

The network linkages discussed in this paper are those between two models of IBM System/360 at the IBM Research Center:

- Model 91 with the System/360 Operating System (OS/360) and two million bytes of main storage
- Model 67 with the System/360 Time-Sharing System (TSS/360) and one million bytes of main storage

The Model 91 is used for high-computation-volume batch applications, whereas the Model 67 provides a powerful, general-purpose interactive service with several features that complement the Model 91. In the installation being discussed, virtual

storage provided by the Model 67 interactive system offers four billion bytes of addressability, whereas our Model 91 is limited to partition sizes of 500K bytes. Further, the higher speed of the Model 91 Central Processing Unit (CPU) complements the slower CPU speed of the Model 67. The Model 91 also complements the greater time required for referencing scattered storage locations in the Model 67 because of its being a paging machine.

The hardware link between the two machines is a 40.8 kilobaud telephone line. The interface between the telephone lines and the processors is through the IBM 2701 Transmission Control Unit.

complementary features

Another complementary feature of the Research Center time-sharing system is a text-editing program that may be used to preprocess data interactively before it is moved to the Model 91 for analysis by programs running under OS/360. Another capability of the text-editing program is that users can convert TSS/360 FORTRAN and assembler-language source programs to OS/360 fixed-column format automatically. The time-sharing system also supports a hierarchial storage system that permits users to keep seldom-used data sets in compressed form in peripheral storage.[1]

There are numerous specialized programming capabilities available under OS/360 that have not been adapted for the TSS/360 interactive system simply because the user population of OS/360 batch system is much larger than that of the time-sharing system. Programs such as sort/merge, the Scientific Subroutine Package, and the Mathematical Programming System would be readily available to interactive system users provided that an efficient data path to the batch system could be implemented.

data interchange

Given the expected advantages, a project was established to develop a data interchange between the Model 67 running with TSS/360 and the Model 91, running with OS/360. This network permits users to move data sets easily from one machine to the other and to request entry into the Model 91 batch-work queue from a Model 67 terminal. Since the version of OS/360 implemented at the Research Center uses the Local Attached Support Processor (LASP) to manage its job stream and output data sets, few programming changes are required on OS/360. The Model 67 is defined to LASP as a remote intelligent terminal, and a network management program—written on TSS/360—conforms to the LASP protocol. Thus the OS/360 user who wishes to send data to TSS/360 supplies an LASP format card that indicates a particular data set to be written into the pseudo device whose counterpart resides on the Model 67.[2]

The major portion of the work discussed in this paper, including all of the performance measuring programming, is related to the

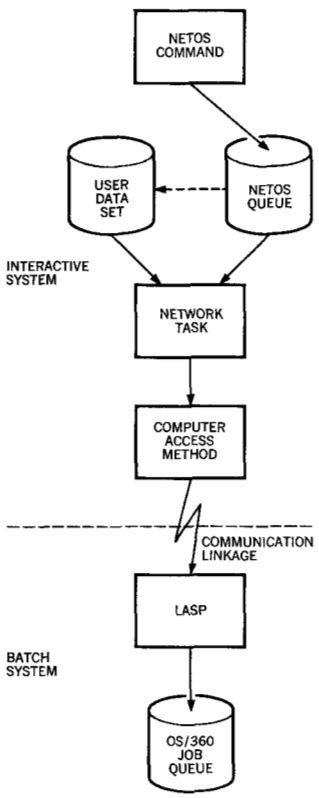
Model 67 and TSS/360. A TSS/360 command has been written that causes a data set that contains an OS/360 job to be sent to the Model 91 for execution. Other commands provide query functions. This paper describes briefly the TSS/360 program that supports the data link, and then concentrates on the measurements taken over a six-month period. These measurements give the usage profile of the data link.

Network operations

commands

Three new commands have been added to the TSS/360 repertoire to manage the TSS/360-OS/360 communication link. A user initiates the transfer of a data set to the OS/360 job queue with the NETOS command, which he enters from his terminal (or as part of a background job). The system takes the name of the data set from the command parameter, then enqueues the request and assigns an identifying Network Sequence Number (NSN). If the user wishes to inspect the status of a pending request, he enters the NETOS? command and supplies his NSN for identification. He enters NETOS? NUMB to query the system about the number of requests in the queue. A user cancels a pending request by entering the CNETOS command, provided TSS/360 has not yet started sending the data set.

Figure 1 Interactive-batch system programming



As illustrated in Figure 1, when the user enters a NETOS command, a record is enqueued on disk storage that indicates the data set name, the owner of the data set, and other information. The request is queued so that the user may continue his TSS/360 interactive work, regardless of the status of the OS/360 batch system or the communication link. A special nonterminating network task manages the NETOS queue and the interactive side of the communication link. When the network task finds an entry in the queue, the task verifies the associated data set and calls on a special table-driven computer access method to send the data to the batch system. On the OS/360 batch side of the network, LASP reads the data and reformats it on disk as a part of the OS/360-LASP job queue.

When the network task has finished sending the data set, it calls on the access method to see if OS/360 has anything to send. Data sets from the batch system are prefixed with a special header record that indicates by a user identification whose job produced the data set. The network task uses this identification to catalog the data set for the proper interactive system user, and then reads the rest of the data from the batch system and writes it into disk storage. When the transmission is complete, the network task attempts to write a message on the user's terminal, advising that his data set has arrived. If the user is not logged on the message is not written.

As long as the NETOS queue has entries in it, the network task alternately calls the access method to transmit a data set to the batch system, then to read from the batch system if it has data to send. Whenever the NETOS queue is empty and there is no data coming from the batch system, the network task is paged out for about thirty seconds. Thus if there is no work for the network task, it doesn't burden the system.

authorization considerations

The TSS/360 interactive system at the Research Center is part of a six-member network of interactive installations.[3] A security problem arises when a path is created between the interactive and the batch system because confidential information may be retrieved from the batch system files via the NETOS link. Therefore, safeguards have been built into the NETOS programming to prevent unauthorized interactive system network users from accessing the Model 91.

Normally the network task is logged on automatically at start-up time and remains active all day. However, it can be shut down or started by operator commands. Thus the operation of the network task is independent of the status of the batch system. Therefore, if the Model 91 is down, terminal users may still enter NETOS commands, and their requests are queued on disk storage. The network task keeps attempting to transmit to the OS/360 batch system until the Model 91 is ready to receive requests.

Data are sent from the interactive to the batch system in card-image format since the data become part of the batch job stream. On the other hand, data sets sent from the batch to the interactive system are not necessarily destined for printer or punch. The program on the interactive side writes the data on disk in variable length records with a maximum size of 947 bytes. The user may print or punch them, but the more sophisticated applications use the data as input to interactive TSS/360 programs. Trailing blanks are stripped from the data before transmission.

Network measurements

An important goal of the network project is the study of the usage of the computer connection. Since few installations have previously coupled a large time-sharing system with a large batch system in this manner, system measurements may be of broad interest. System measurements include the following:

- Frequency of NETOS executions
- Types of data sets transmitted
- Classifications of communication line error recoveries
- Amount of time that requests are delayed in the NETOS queue

To this end, a continuous log of "time-stamped" entries is kept of

NETOS activity. Thus, for example, one can determine when a user enters a NETOS command, how long it remains in the queue, how many bytes are in the data set, and how much time is spent opening the data set, reading it, and transmitting it to the Model 91.

network utilization

The interactive-batch network has enjoyed increasing usage at the Research Center since it began in late 1970. Figure 2A shows growth in the number of users; Figure 2B shows the number of NETOS commands entered; and Figure 2C shows that the number of bytes transmitted to the Model 91 continues to rise as new uses are discovered for the data link. There are over eighty people who have used the services of the netting task at some time. During a typical week, about thirty people—twenty percent of the average weekly user load—are found to enter NETOS commands or send data from the batch to the interactive system with a format card. During that period, an average of two to three hundred commands are entered. The quantity of data sent from the terminal-oriented system to the batch system fluctuates from day to day. It has run over forty million bytes in a five-day week, but the average weekly quantity is about sixteen million bytes. High variation is also apparent in the amount of data sent from the batch to the interactive system. The netting task generally handles about three million bytes per week, representing thirty to sixty data sets.

transmission statistics

As use of the communications link increases, the implementers are continuously monitoring the amount of time required to service NETOS requests. The average data set size is about forty-five thousand bytes, with a resulting transmission time of about ten seconds. Since requests are queued on disk, and since there is additional overhead in accessing the user's data set, the actual service time is somewhat longer. Nevertheless, about fifty-five percent of the requests are serviced within two minutes, and forty percent are handled within sixty seconds of entering the command, as indicated in Figure 3. Such waiting times are quite acceptable, especially since they include time during which the batch system is down, when the network task is not active, and when the interactive system goes down between the time a NETOS command is entered and the data is transmitted to the batch system.

Also monitored are the hour-by-hour transmissions to and from the OS/360 batch system—as shown in Figure 4—to see if any overload periods become apparent. As expected, NETOS usage follows the profile of the overall interactive TSS/360 load, i.e., most commands are entered in the late morning and the middle of the afternoon. Data set transfers from the batch OS/360 to the interactive TSS/360 system are more evenly distributed throughout the day, since random service times on the batch system tend

Figure 2 Growth in usage of the interactive-batch network for 1971

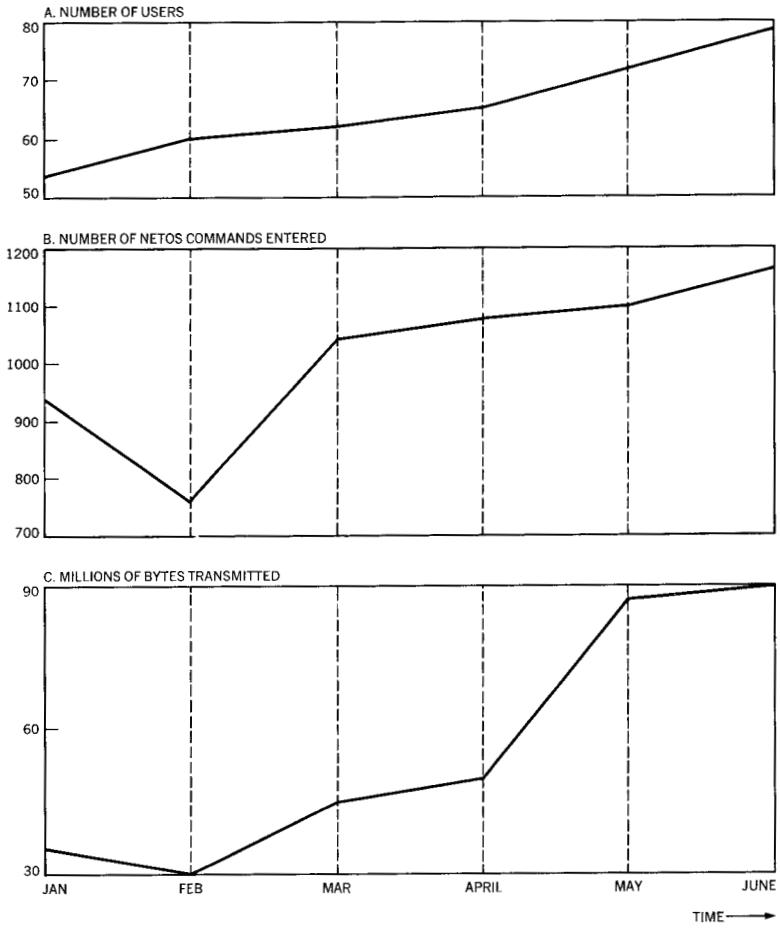


Figure 3 Percentage of NETOS queue entries as a function of time in the queue

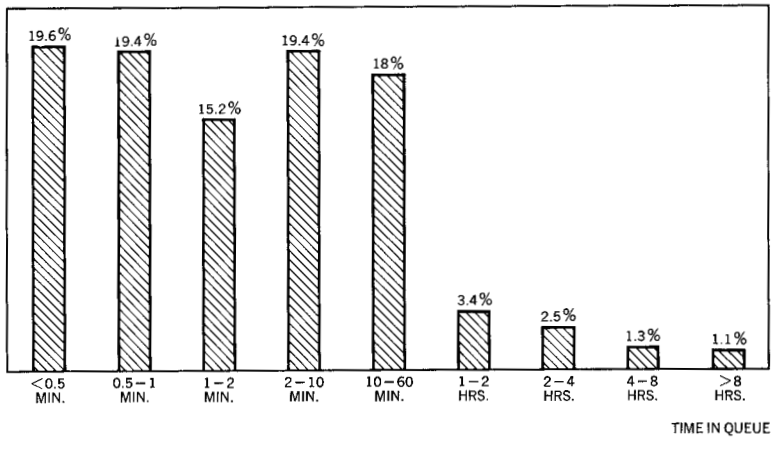
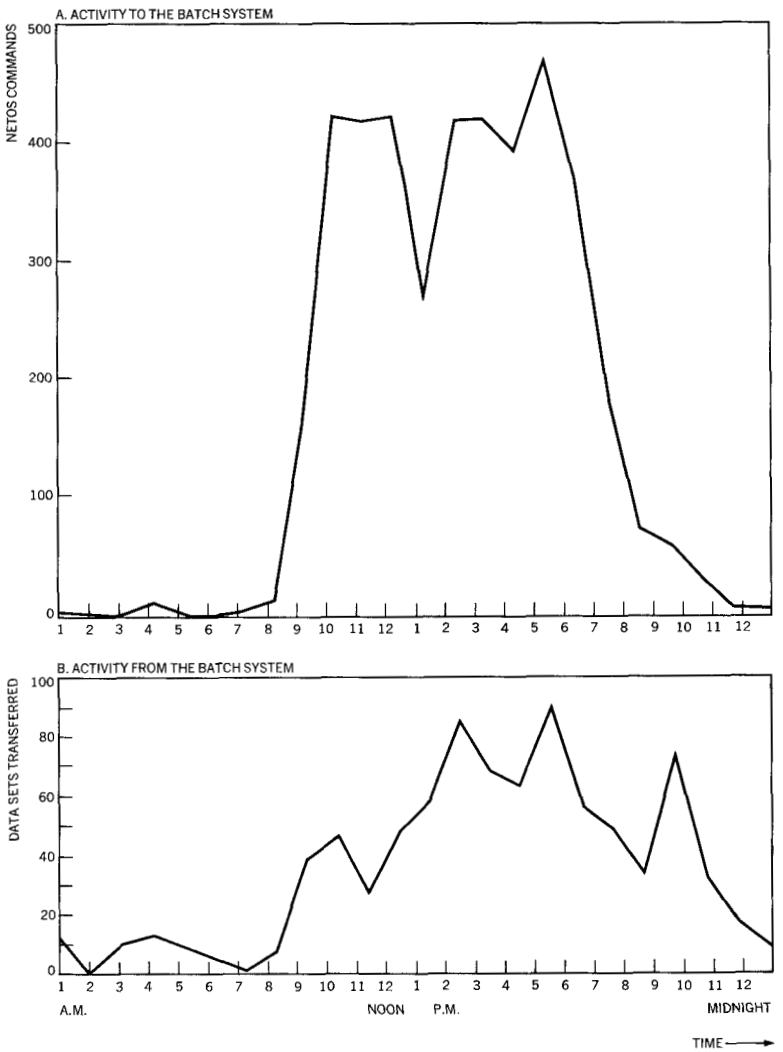


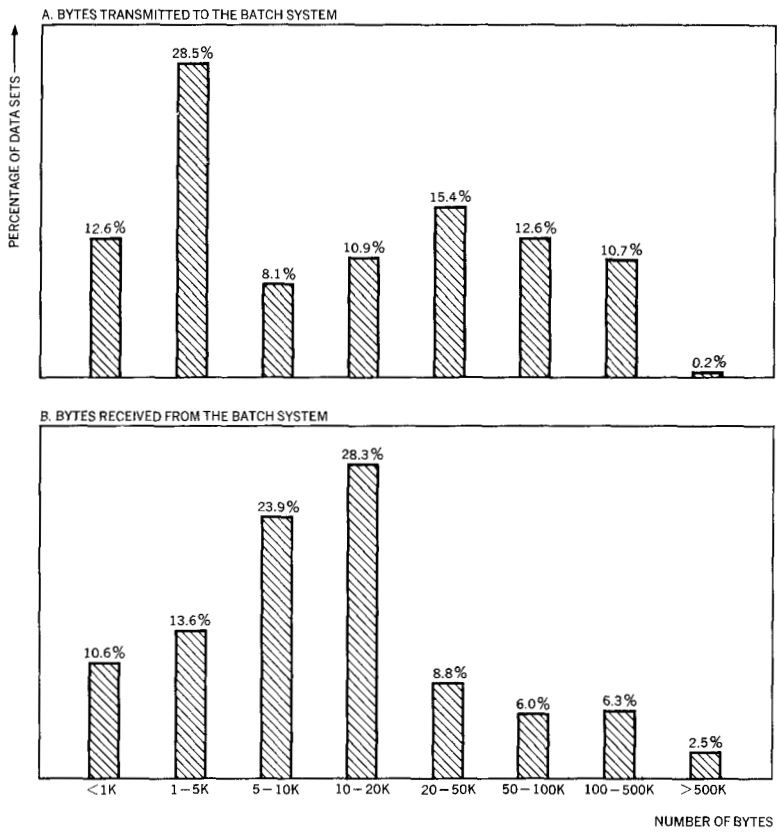
Figure 4 Network usage



to even out the bumps. There is, however, a heavy load from the batch system around five o'clock p.m. that is being carefully monitored for the following reasons. Data from the batch system takes priority over data to that system. In fact, a transmission from the batch system can interrupt data being transmitted to that system. Although it has not yet occurred, the five-o'clock rush from the batch system could create long waiting lines on the interactive system.

Another aspect of the network that is being observed is data-set size, which is summarized in Figure 5. As mentioned previously, the average data set size is about forty-five thousand bytes. Monitoring measurements indicate that data-set sizes are bimodally

Figure 5 Data set size



distributed, with concentrations in the one-to-five-thousand byte range and in the twenty-to-fifty-thousand byte range. Although it is unfeasible to analyze the content of data sets processed by the netting task, a reasonable speculation is that data sets under five thousand bytes consist principally of Job Control Language statements belonging to the programs that use data sets resident in the batch system. The larger shipments may be those jobs whose data resides on the interactive system and which are shipped along with the Job Control Language statements. There has been an increasing trend in the percentage of data sets larger than twenty thousand bytes. Hopefully this is due to growing user confidence in the network programming that is leading users to transfer data bases from the batch system to take advantage of the hierarchical storage of the interactive system. After being transferred to the interactive system, such data sets are sent to the batch system as part of the user's job.

Statistics are also kept of transmission line problems, which show a very clean communications interface. The access method finds between sixty and three hundred I/O faults per week with a mean

of one hundred sixty. These are primarily interactive system timeouts, which occur when the Model 91 is down or when the batch system operators have deleted their side of the communications link. Another type of fault—though not strictly an I/O error—occurs when the batch system interrupts the transmission of a data set from the interactive to the batch system. Recall that the interactive system occasionally suspends transmission because the batch system has precedence over the interactive system in certain phases of line control. These suspensions average eight per week, on a very erratic basis. (During some weeks there are no suspensions, and during one reporting period there were thirty-three out of ninety-six days that were free of this type of fault.)

System implications of the network

TSS/360 runs with a *table-driven scheduler*, which is a set of controls used to govern the execution of tasks. Using the table of twenty-six parameters that direct the frequency and length of time a task is allocated the CPU, the scheduler may be fine tuned by adjusting the parameters.

Of these parameters, we may consider three that are of particular significance to the computer netting task—*priority, delta-to-run*, and *maximum number of pages*. Priority determines the position a task assumes in a list of dispatchable tasks. Low-numbered tasks have the highest priority. These tasks head the list and are serviced first. Delta-to-run (DTR) is a factor that is used to determine how long a task is allowed to wait for dispatching before it is eligible for preferential treatment. Maximum number of pages is that number of pages a task may have in storage at a given time.

The adjustment of the scheduling parameters is often done on a trial-and-error basis, the objective being that of dispatching tasks often enough, yet not so frequently that other tasks in the system are degraded. After the parameters of our system had been initially set, three fine tunings were required to bring it to its current operational status. These adjustments of the schedule-table levels (values of the parameters) were an important part of installing the computer network task because it is a highly active program that is critically time dependent.

initial parameters

When the network task was first installed, its parameters were set to resemble a background task: DTR 2.5 minutes; priority 15; and a maximum of 32 pages. This level was unworkable from the start. As the interactive system became loaded through midafternoon, the network task was not executed often enough to support communications. For example, the batch system would send an

inquiry, and then there would be a time out before the interactive system was given control to send a response. The result was that the computer access method and LASP played an endless game of error recovery.

first system tuning

Since it is a relatively easy matter to juggle the TSS/360 schedule table, the error recovery tug-of-war was quickly halted. The DTR was set to zero so that the network task was always behind time and hence given preferential scheduling treatment. Thereafter, the network task was able to provide faultless communication support. The system as a whole, however, was degraded unbearably. Any time the network task was active, all other interactive users had to share a substantially diminished CPU.

second system tuning

Two actions were then taken to reduce the impact of the network task. By activating the task less often, it became inactive for a longer period of time if there were no pending transmissions. Originally, the inactive period was ten seconds, which was arbitrarily selected to serve the network queue adequately. The period of inactivity was increased to thirty seconds, so that the network task was dispatched only one-third as often when the queue was empty.

The second action was to reset the DTR to about 2.5 seconds, and to decrease the priority from 15 to 14 (i.e., to a higher priority). The DTR of 2.5 seconds was selected on the basis of transmission-control considerations. Because the IBM 2701 Transmission Control Unit has a three-second time-out, the network task must gain control within that period to support its side of the communications line. After these changes, the network task ran properly. On the other hand, the system as a whole remained somewhat degraded.

third system tuning

The third tuning—which was a hybrid maneuver—set the system's current operational characteristics. Hardware time-out on the transmission control unit has been adjusted to 10 seconds, and the DTR of the network task is set at 8.5 seconds—which satisfies all the constraints. When the network task goes behind schedule, it still has 1.5 seconds before a hardware time-out, and the system is not affected adversely.

Main storage usage, previously mentioned as a schedule table parameter, was also adjusted during the system-tuning period. Initially, the main storage maximum was set at thirty-two pages. Concurrently with the first schedule-table change, the network task was given the following two schedule-table entries with identical parameters except for the maximum number of pages: one entry is for a thirty-two page maximum and the other for a twenty-four page maximum. The network task thus occupies one of two schedule-table levels. The task is initiated in the schedule-

table level with a maximum number of pages of twenty-four and is moved to the thirty-two page level only if necessary. Experience shows that the majority of dispatches are at the twenty-four page level. Results of a typical day show 9512 occupancies of twenty-four pages and only 116 at the thirty-two page level.

Both the network internal statistics and TSS/360 global-use tables show that the usage of the network task has grown rapidly since it was first installed. TSS/360 keeps track of the number of times a task is scheduled for a time slice, thereby giving a reasonable measure of task activity. When the network was first installed (and after the final tuning of scheduling parameters) the network accounted for about four percent of total dispatches. After six months of operation, network dispatches were up to ten percent of total dispatches, and the network was the third most active schedule-table task.

system experience

What does this computer network experiment mean in terms of impact on the system? Of course, with an interactive paging system such as TSS/360 it is difficult to place a numerical value on system impact. The network task went on the active list ten percent of the time. Not known, however, is how much of its time slice was used, although it is known that the network task did not use up ten percent of the total CPU time. A good measure of the impact of a new major subsystem is its effect on the system monitor. In the case of this experiment, after the tuning period, the system monitor response time was down to a uniform 1.5 seconds, which represents a very little impact on TSS/360 by the network task.

The key point here is that the network task had the potential for impacting the system heavily. This is evident from the fact that when the system had to service the transmission control unit within the three-second time-out period and transact its own internal business, the network task could and did at first cripple other tasks.

Network experience

There are several factors that contribute to the usefulness of the TSS/360-OS/360 linkage. First, the two systems complement one another. OS/360 on the Model 91 yields very high throughput, whereas TSS/360 serving over forty simultaneous users cannot supply the high computing power needed for many research problems. However, the interactive TSS/360 system, augmented by several locally developed facilities, provides excellent interactive support and data management facilities. The interactive editor permits the user to alter data sets before and after transmission to the OS/360 batch system. TSS/360 also supports several levels

of direct-access storage hierarchy not available on OS/360. The data may reside in online public storage (in packed or unpacked form), or may be committed to peripheral storage, which exists in easily accessed packed form on disk. Thus a user of large data bases may find it economical to keep his data on TSS/360 public storage and send it to OS/360 for batch processing.

The success of the computer network also depends on the speed with which one may enter the OS/360 job stream. The transmission time is negligible, and time spent in the NETOS queue is typically less than a minute. In comparison, a user finds it easier to enter a command through his terminal than to transport a box of cards to the Model 91 card reader. While the user is logged on, the system notifies him automatically when his data set(s) have been returned to the time-sharing system, provided he has requested such action with a LASP format card.

Another factor favoring the use of the network is the ease with which satellite processors such as the IBM 1130 and IBM 1800 can be managed interactively through TSS/360.[5] Several applications in the Research Center depend on 1800s or 1130s as data collection stations. Because the small processors are not powerful enough for data analysis in some cases, computing services of the Model 91 are often required. Though programs exist for sending data over telephone lines from a satellite computer to the Model 91 directly, the procedure is rather involved. (In the batch environment, the computer operator must be called on to release a job before communications can be established.) On TSS/360, however, the user can coordinate use of the communications link from his own terminal. Since the Model 91 can be readily accessed by the NETOS command, several users prefer to go through TSS/360 to transfer data to the batch system.

The computer network programming provides a unique multiprocessing capability, and at least one researcher is experimenting with the data link in this way. His TSS/360 program builds an OS/360 job that is sent to the Model 91 by means of the NETOS command. While the OS/360 job is in execution, a TSS/360 program is computing in a true multiprocessing mode. Thus two physically separated and logically distinct computing systems are working simultaneously on different parts of the same problem. When the TSS/360 program is ready for the output from OS/360, the interactive program either waits until the batch-processed data set arrives, or operates on the data immediately if batch-processed data is available ahead of the interactively computed data. Thus we have created a multiprocessing system using machines whose CPU speeds and storage management differ significantly without resorting to highly sophisticated system programming techniques. This same system uses the TSS/360-1130 communications software. Triplexed multiprocessing between

the IBM 1130, the Model 67, and the Model 91 offers challenging opportunities for future research.

Several possible extensions to the computer network that have been suggested are now discussed. Currently, the user can check the status of his interactively entered request up to the time the data set is sent to the batch system and enters the job stream. Thereafter, he must use a console in the computer center if he wishes to learn the status of his job. It would be convenient if TSS/360 could determine the status of the job from OS/360. Then the user could check the status of his request even after the data had been sent to the batch system. LASP does not support such a remote inquiry feature, although it is offered under new versions of HASP.

A primary fault of the present system—also due to LASP limitations—is that records are not interleaved; that is, a data set must be sent as one integral transmission. Thus if the batch system has a very large data set to return to the interactive system, it may hold up the NETOS queue since no data can be sent to OS/360 until its transmission is complete. A better design would be to interleave transmission blocks when there is data to be sent from both systems.[6]

Concluding remarks

The computer network communications programming discussed in this paper allows users to combine the best aspects of two powerful systems. TSS/360 can be used as a base of operations, taking advantage of the hierarchical storage facility. Data sets can be altered interactively using the editing program, then quickly sent to OS/360 via the NETOS command. Users who work primarily on TSS/360 can take advantage of special features on OS/360 such as language processors or utilities and have the resultant data sets returned and catalogued automatically. Travel time between a scientist's office and the computer installation is minimized, and a user knows instantly when a job has finished, provided data sets are returned to TSS/360 while the scientist is logged on. If they desire, users can enter programs into the Model 67 to be executed in the background.

As indicated by its measured usage, the computer network has become an important part of the overall architecture of the Research Center computer complex. New users are added each week. New uses are discovered where none were suspected—such as the asymmetric multiprocessing application. At the same time, the impact on TSS/360 as a whole is minimal, since priority levels for the netting task may be easily adjusted via the table-driven scheduling facility.

The system has attracted a level of use that warrants extension of the concept to other systems. For example, it may be possible to implement a similar facility at the Research Center between CP-67/CMS[7] and OS/360. Such a data link could be used for further parallel operations, since CMS and OS/360 use a compatible object code as well as compatible FORTRAN and assembler source codes.

Because TSS/360 and OS/360 each have features that complement one another, they are ideal systems to link together. Experience has shown that when a data path is available between complementary systems, it is exploited fully. The network discussed in this paper, although heavily used, is far from being saturated.

ACKNOWLEDGMENTS

The NETOS programming was developed jointly by the author, J. W. Meyer, and R. Nachbar of IBM Research. The author wishes to acknowledge the contributions of A. H. Weis and W. J. Doherty also of IBM Research, and of R. Seymour of the Bell Telephone Laboratories, Naperville, Illinois.

REFERENCES

1. J. P. Considine and A. H. Weis, "Establishment and maintenance of a storage hierarchy for an on-line data base under TSS/360," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **35**, 433–440, Thompson Book Company, Washington, D.C. (1969).
   See also A. M. Katcher, "Efficient utilization of limited access archival storage in a time-shared environment," *Proceedings of the ACM Symposium on Information Storage and Retrieval*, April 1971, College Park, Maryland, Published by the Association for Computing Machinery, New York, New York (1971).
2. *IBM System/360 Attached Support Processor System (ASP), Version 2, System Manual*, Form GY20-0305-0, First Edition (December 1968). International Business Machines Corporation, Data Processing Division, White Plains, New York.
3. R. M. Rutledge et al., "An interactive network of time sharing computers," *Proceedings of the 24th National Conference, Association for Computing Machinery*, San Francisco, California, August 26–28, 1969.
4. *IBM System/360 Time Sharing System, System Logic Summary, Program Logic Manual*, 110–113, Form GY28-2009 (June 1970), International Business Machines Corporation, Data Processing Division, White Plains, New York.
5. W. S. Hobgood, "Software support for satellite processors," IBM Research Report RC 3372 (July 1971). May be obtained from the author: IBM T. J. Watson Research Center, Yorktown Heights, New York 10598.
6. *IBM System/360 and System/360 Attached Support Processor System (ASP), Version 2, System Programmer's Manual*, Form GH20-0323, 27 and Appendix G (March 1971), International Business Machines Corporation, Data Processing Division, White Plains, New York.
7. R. A. Meyer and L. H. Seawright, "A virtual machine time-sharing system," *IBM Systems Journal* **9**, No. 3, 199–218 (1970).