This paper discusses the evolution of a facility, generally called job networking, that permits job-related information to be sent between programming system components operating on computing systems that are attached to a communications network. The capabilities of the programs that implement this facility are described along with the events that led to its development, the value of the facility to the user, and ways in which it can be extended to work with other forms of communication networking.

Job networking

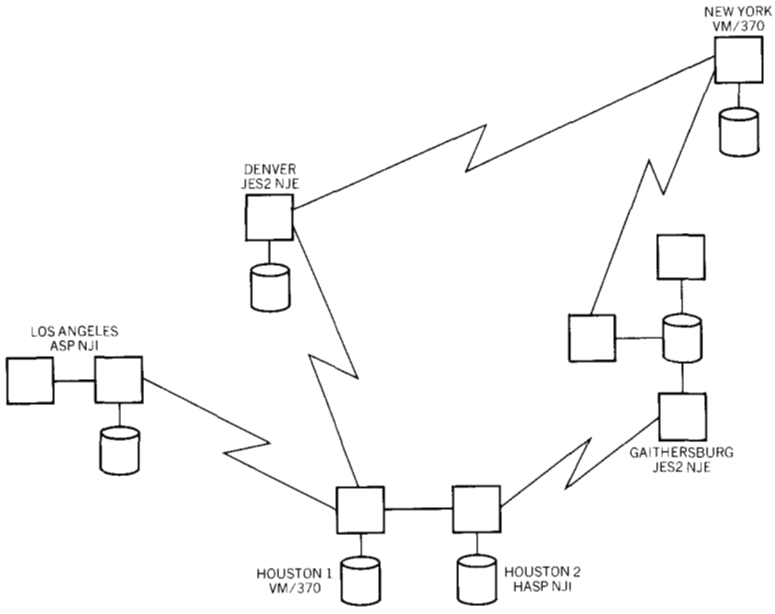
by R. P. Crabtree

In 1976 IBM announced a number of new systems programs for data communications. These programs, called the Advanced Communications Function,[1] introduced new capabilities in Systems Network Architecture,[2] especially in the area of multisystem communication (or networking).

At the same time, four programs were announced in the area of job entry subsystems which, while not part of the Advanced Communications Function, were also related to networking. The Network Job Entry Facility for Job Entry Subsystem 2[3,4] (NJE for JES2) offers installations using Operating System/Virtual Storage 2 (OS/VS2) with Multiple Virtual Storage (MVS)[5] the capability to transmit selected jobs and in-stream data sets, system output data sets, operator commands and messages, and accounting information from one JES2 MVS computing system to another across a communication link.

Network Job Interface (known as the NJI PRPQ[6]) provides a similar capability for three other types of computer installations by offering modifications to the programming systems used in these installations: For installations using OS/VS2 Single Virtual Storage (SVS) and the HASP system, the HASP Networking PRPQ[7] was introduced. For installations using the Asymmetric Multi-

Figure 1 Possible NJE/NJI configuration



processing System (ASP), Version 3.2, with OS/VS2 SVS, the ASP Networking PRPQ[8] was introduced. And finally, for installations using the IBM Virtual Machine Facility/370 (VM/370), the VM/370 Networking PRPQ[9] was introduced.

The three NJI programs provide capabilities similar to NJE for JES2 and are compatible not only with each other but also with NJE for JES2. As such, they provide a way to establish a communications network between the job entry components of any combination of JES2 MVS, HASP SVS, ASP SVS, and VM/370. Figure 1 shows a possible NJE/NJI configuration consisting of all of these new job entry programs. The resulting network, which permits job-related information to be sent between any of the attached systems, can be called a job network, and the process of using the capabilities offered by these programs is usually referred to as job networking.

The job entry component of each system is responsible for the entry (receiving), scheduling, and output (printing and punching) of jobs, for interfacing with the computer operator for job control functions, and for a substantial portion of the job accounting function. Some job entry subsystems, for example ASP and JES3 (Job Entry Subsystem 3),[10] provide other additional functions such as premounting required volumes before a job is run, or scheduling the execution of jobs in specific interdependent sequences.

Unlike the other operating systems mentioned here, VM/370 is oriented more toward the concept of "interactive computing" rather than "batch jobs"; as such, VM/370 has no specific job entry subsystem. Nevertheless, its networking component does have many of the same functions and responsibilities as a job entry subsystem, and it provides many of the same services or benefits to the network.

This job networking capability is not entirely new to IBM programming systems. ASP (and later JES3) included a feature called network job processing (or NJP) that allowed jobs to be sent between ASP systems (or between JES3 systems), but this feature proved to be too restrictive for many installations to be able to use it in a production environment. Thus, these new job networking products can be thought of as introducing new functions to IBM operating systems.

In the following sections, the characteristics and value of job networking are discussed, followed by a description of the events in the development of job networking programs and some uses of job networking.

Characteristics of job networking

For the purposes of this discussion, communication can best be divided into three basic types:

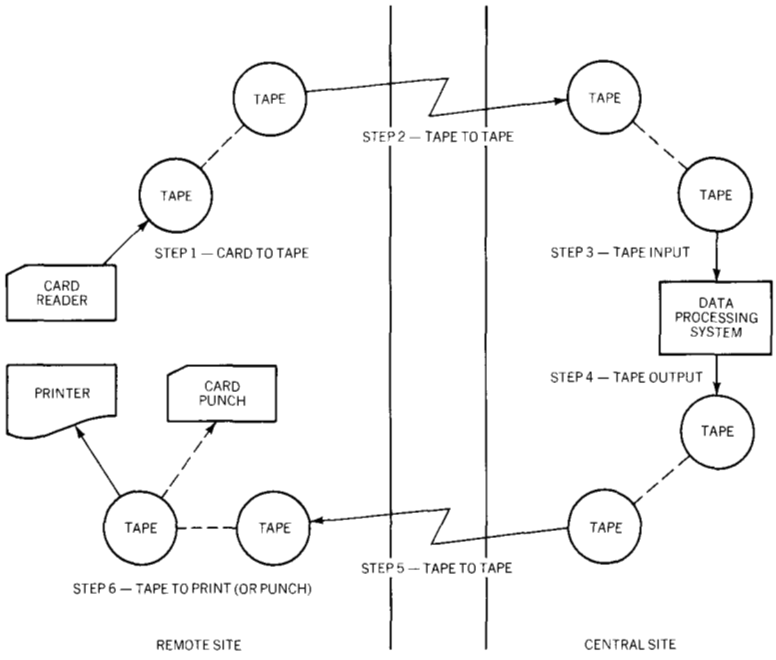
1. Message switching.
2. Interactive.
3. Batch.

Message switching communication is characterized by transmissions consisting of relatively small messages with little requirement for a rapid response. Interactive communication also usually involves relatively small messages (low volume per transmission) but, unlike message switching, usually includes a strong requirement for a fast response. The purpose of this discussion primarily concerns the third type, batch communication.

Batch communication usually involves transmissions with relatively high volumes and little requirement for a fast response. Historically, batch communication has been represented by three different types of implementation:

1. Remote job entry (RJE).
2. Bulk data transfer.
3. Job networking.

Figure 2 Using bulk data transfer to accomplish rudimentary RJE



Remote job entry facilities have been in existence for well over ten years and are very basic to today's large computer installations. The primary purpose of remote job entry is to extend the support provided for local unit record devices (card readers, printers, card punches, and consoles) out to a remote location through the medium of a communication line.

Remote job entry protocols are usually very device-oriented and tend to be very simple in order to minimize the intelligence required in the remote controller. Although basically batch-oriented, remote job entry must interface closely with a remote operator and usually offers him an interface to the system to enter commands and receive messages in response. This interface tends to be more interactive than batch-oriented, and the remote job entry protocols must also address this requirement.

Bulk data transfer applications go back even further than remote job entry. Not only have numerous applications been written to accomplish this function, but special devices have been manufactured and marketed (e.g., the IBM 7702 and 7711 Data Transmission Units) to perform this function without the requirement for programming support.

In the early days before remote job entry was available, bulk data transfer applications and devices were sometimes used to accomplish the same function. Figure 2 shows how this transfer was

achieved in many installations. Jobs were transcribed to tape at the remote site, and then the contents of that tape were "transferred" to another tape at the central site by a bulk data transfer application or device. This duplicate tape was then used as an input tape to the data processing system.

For output functions the process was reversed. Job output would be transcribed to tape by the computer, and this tape would be "transferred" to the remote site where its data could then be printed on paper or punched into cards.

Compared with bulk data transfer and remote job entry, job networking is relatively new. This form of batch communication is defined as a facility for transmitting jobs (consisting of job control language statements and embedded in-stream data sets), system output data sets, job-oriented commands and operator messages, and job accounting information from one CPU to another.

In the simplest form, two processors connected by a communication link could constitute a "job network." A job could enter the network through any local card reader, remote job entry station, or internal interface at either CPU. Once entered, it could either be executed locally or be transmitted to the other CPU for execution. Upon completion of execution, the output could be printed or punched on local devices or remote job entry stations attached to the execution CPU or could be transmitted to the other CPU for output processing on its local or remote output devices.

In a more general job network consisting of many processors and communication links, jobs could be submitted anywhere in the network and routed to any processor for execution. Likewise, the output generated by any job running anywhere in the job network could be routed to any output device or group of output devices in the network for output processing.

Value of job networking

In order for a product to be accepted and used, it must offer some value to the user. In this case, a number of functions can be performed that result in tangible benefits to the enterprise establishing a job network. More specifically, a job network allows an enterprise to perform the functions discussed below.

move jobs

With a job networking capability, a job can be entered into the job network at any location and be transmitted to the location having the data sets that are required to successfully run the job. Perhaps the job requires the brute processing power or the large storage of a high-speed, large-capacity computer located at another site rather than the smaller computer that may be available locally.

Or, a job may be transmitted to a location that has a special processor hardware feature (such as an emulator) or a special hardware device (such as an array processor) that is not available on the local system.

In some cases, specific applications are only supported at certain sites within an enterprise. Job networking permits jobs to be routed from other sites in the enterprise to the system where an application is supported. In other cases, the customer might be able to more easily justify the cost of a licensed program by collecting jobs that use the program from all over the job network.

move job system output

The most obvious reason for transmitting a job's system output data sets is to get them to their appropriate locations. For example, a report program may be run, and copies of its output may be distributed automatically via the job network.

System output data sets may be sent to locations having special output equipment needed by the data sets. An example is a printer that can print reports using multiple character sets on the same page. The output device need not be a printer or even a card punch, however. A microfilm recorder or plotter may be the desired destination.

assist migration

The ability to connect CPUs together provides a vehicle to migrate from one hardware or software system to another. As new products become available, an enterprise has the option of installing the new products at specific points in the job network. Selected jobs can then be routed to the new product to accomplish an ordered transition to the new product. The migration to be accomplished may be as minor as the transition to a new level of an existing application, or it may be as major as the conversion to a new type of operating system.

History of job networking

ASP network job processing

The first IBM program to support job networking appeared late in 1969 with the release of the network job processing feature of the ASP system. This implementation of job networking had essentially four flaws that prevented it from being generally used throughout the data processing community.

First, it used the basic terminal access method (BTAM) with binary synchronous communication (BSC) links and protocols. This arrangement provided a communication medium limited to half-duplex, unidirectional, nonmultiplexed transmissions that was unable to provide the characteristics required to properly support job networking.

Second, network job processing did not provide any routing support through the job network and required that the submitter of the job provide (through control cards) explicit directions as to how the jobs and system output data sets should be routed through the network. The user was burdened with the responsibility of being aware at all times of the topology or configuration of the job network.

Third, limitations in operator commands made the system extremely difficult to use. Since the routing of a job was explicitly specified in the control cards accompanying the job, it was difficult to change the routing once the job was submitted. If a particular link that a job had specified was not operational, the job would not be able to get to its destination even though another path might be available.

But the greatest restriction resulted from the technique of transmitting ASP control blocks from CPU to CPU. This technique not only required all CPUs to run ASP, but the same version and level of ASP. An ASP program change that modified the format of one of the transmitted ASP control blocks would have to be applied to all systems simultaneously in order to continue job networking successfully. If one of the CPUs in the job network were to install a new version or level of the ASP system, all other CPUs in the network would have to upgrade to the new system at the same time if the job network were to remain useable.

early customer implementations

Around the same time, a number of customers started making modifications to the HASP system to implement job networking in that system. These modifications were freely exchanged and distributed through the various user groups such as SHARE and GUIDE and exhibited the following advantages over ASP network job processing.

First, they used the remote terminal access method (RTAM) with MULTI-LEAVING protocols that provided a communication medium with bi-directional and multiplexed transmission capabilities. This meant that several jobs and system output data sets could be transmitted concurrently along with operator commands and messages in both directions at the same time.

Second, a small number of commands were implemented to query and control jobs and system output data sets from distant CPUs. Although the commands were quite limited, the capability was there and could be expanded by the individual installation.

Third, and most important, the unit of communication was the input or system output record, the operator command, or the operator message rather than the internal control block. This tech-

nique provided independence to each system in the network and allowed each to migrate to new versions and levels of HASP without regard for the other systems on the network.

The primary limitations of these early HASP implementations included the lack of any network control and the inability of systems to forward jobs and system output data sets once they had received them from another member of the job network. There was also a lack of accounting information generated by these early user systems.

development of NJI

In early 1973, IBM was attempting to solve some of the synchronization and communication problems it was experiencing in developing related system components in laboratories scattered throughout the world. As a solution to these problems, a number of modifications to the systems that IBM was using in its internal computing facilities (HASP, ASP, and VM/370) were developed. At first, these modifications were informally developed and no clear structure was obvious. By 1974, however, a clear pattern was beginning to emerge from these modifications as the resulting facilities being developed began to look more and more like job networking.

By April 1974, there were two separate job networks operating within IBM with a total of over 25 processors involved. Representatives from the various IBM internal sites met and generated specifications that would not only allow the two discreet networks already in existence to communicate with one another, but would also allow the then soon-to-be-released new operating system, MVS, to be modified in a similar fashion to interface with the existing systems. At this time, the internal network was named "SUN" for Subsystem Unified Network.

The SUN system shared many of the characteristics of the user-developed systems in that it used RTAM and the MULTI-LEAVING protocols for communication; it included operator commands and messages for control and used record level transmission blocks for maximum system independence. But the SUN system also supported the "store and forward" capability, that is, the ability to receive a job, data set, command, or message from one processor in the network and pass it on to another; and it also introduced an expanded interface with interactive users (Time Sharing Option and Conversational Monitor System) to allow them to interface with the job network and receive information from it.

In November 1974, this SUN network was described to the Computer Network Project of the SHARE user group in an attempt to get some feedback that could be used as input to the job network-

ing development being started in the JES2 development group. As a result of this presentation, the attendees began to request that IBM release the SUN modifications as a product.

Since the SUN system had not been developed for release, it took a great deal of effort and time to pull together the various modifications and package them in a form that could be used by a customer. In November 1976, IBM announced that the modifications would be made available. By this time, the modifications were separated into three separate products, and the package was named Network Job Interface to stress the true nature of the modifications and their compatibility with the network job entry facility for JES2.

development of NJE for JES2

As the development of NJE for JES2 progressed, a great deal of attention was paid to the progress of the SUN network. As the network grew, more and more effort had to be expended by the system programmers at each installation to keep the individual nodes aware of the network topology or configuration. It seemed to the NJE implementers that most of this effort could be managed by a component that was part of each NJE processor.

This concept led to the most unique component of NJE, the "network path manager." With this component, the system programmer is not burdened with specifying the job network topology. The programmer simply must specify the number and names of the nodes and the number of local lines to be used for job networking; the network path manager does everything else. By communicating with its counterparts in each adjacent processor, every NJE node keeps a picture of the current configuration of the job network and makes routing decisions based on this picture. Thus, NJE is capable of adjusting dynamically to any change in the job network configuration just as long as the three basic characteristics specified by the system programmer do not change.

To allow for the fact that systems without network path managers may wish to attach themselves to an NJE network, special provisions were made to allow "predefined connections" to be specified. These provisions allow NJI systems to become fully qualified members of an NJE network.

The network path manager also allows NJE to support parallel links and multiple paths. Parallel links are communication links that exist in parallel between any two NJE nodes. The NJE path manager supports any number of parallel links between any number of adjacent nodes. Multiple paths exist when there is more than one path through the job network between the origin and destination nodes. The NJE path manager will support any number of multiple paths in a job network and will effectively use up to eight of them simultaneously if the characteristics of the paths

allow it and the workload is sufficient. In effect, the network path manager allows NJE to support any practical interconnection of NJE nodes.

Another problem that confronted the NJE implementers concerned the multiaccess spool configuration of JES2—a configuration that does not apply to any of the other job networking programs. In this configuration, more than one (up to seven) JES2 system can share a common spooling facility (the composite direct access space used by JES2 for the storage of jobs, in-stream data sets, and system output data sets) and associated job queue. Any system can add work to or remove work from the common job queue as jobs, system output data sets, or resources become available. Unlike the ASP and JES3 configurations, there is no master/slave relationship between the systems, and more than one system might be connected to the job network. Just how to look at this configuration from a job networking standpoint was a major consideration.

If the entire configuration was viewed as one "node," then jobs and system output data sets could be simply routed to the common node. Console messages, however, would have to be sent to the individual system to which the designated console was attached. This arrangement would require that a two-level routing mechanism (i.e., node and system) be designed and supported throughout the job network.

Conversely, if each system in a multiaccess spooling configuration was considered to be a separate node, then each member of the job network would have to realize that all of these nodes represented the same job queue and that no distinction should be made when routing jobs or system output data sets. The NJE systems that were part of the common pool would have to accept work routed to any of the nodes with the understanding that it was for the common job queue and could be processed on any member of the multiaccess spool configuration. The user and system programmer also would be faced with multiple node names to worry about and with the increased complexity caused by this philosophy.

In the end, it was decided that the entire multiaccess spooling configuration would be treated as one node, and the associated dual-level routing mechanism for console messages was designed and implemented, not only in NJE for JES2, but in each of the NJI components. In addition, the JES2 job queue was expanded to allow communication between the members of the multiaccess spooling configuration, permitting the network path managers and console processors in the various members to communicate with one another. This technique effectively added the shared direct access storage device to the list of communication links that NJE for JES2 supported.

All other subsystems for which job networking had been implemented restricted the size of the spooled record to 256 or less bytes. But JES2 allowed records with lengths up to 32,767 bytes to be spooled, and NJE for JES2 was also required to be able to transmit and receive these long records. This greater length was a problem because it meant that the logical length of the records could greatly exceed the physical length of the transmission block. To solve the problem, the design for NJE included a segmenting scheme to break these long records into smaller ones that could be more easily processed. When received, these records are assembled back into long records with their original logical record length.

An extensive set of operator commands was implemented for NJE along with a concept of "global" commands. Global commands are transmitted as job networking control blocks rather than as they were entered from the console. By interpreting these control blocks, any member of the job network can respond to these global commands regardless of the local console format of the command. Thus, an operator is permitted to inquire about and control work in another node in the job network without having to understand the operator command syntax at that node.

It was felt that it was unreasonable to require each node in the job network to use the same communication buffer size. Each node would have its own requirements based upon the type of remote job entry terminals that it was required to support, and those requirements should not preclude connection to the job network. For that reason the network path manager was designed to exchange buffer sizes with the network path managers in each of the adjacent nodes. NJE then uses the smaller of these two buffer sizes for communication between the two systems.

Various other additions were made to NJE for JES2, particularly in the area of job and network accounting. Provisions were made for additional accounting fields and user exits to permit tailoring the information collected and produced to meet the accounting needs of the installation.

Further developments in job networking

The MULTI-LEAVING protocols that NJE uses offer several advantages over the basic binary synchronous protocols used by the first implementations of job networking. Still, the MULTI-LEAVING protocols themselves are limited in that they are built upon binary synchronous links that are inherently half-duplex (only able to transmit in one direction at a given time) even though the communication lines themselves are often able to support full-duplex operation.

The use of RTAM also produces limitations in that it requires that the communication link, when in use, be dedicated to NJE. If other applications in the NJE processors wish to communicate, they must either establish other communication links or wait until the NJE communication links are shut down.

The use of RTAM and the MULTI-LEAVING protocols also create performance considerations when jobs or system output data sets have to be forwarded through an intermediate node. The MULTI-LEAVING implementations require that the entire job or system output data set group be received and spooled at an intermediate node before transmission to the next node can begin. This requirement not only increases the time for a job to pass through the network, but increases the CPU and spooling requirements at the intermediate node.

All three of these limitations can be removed by using the Advanced Communications Function of the Virtual Telecommunications Access Method (VTAM) and the Network Control Program (NCP). By interfacing with VTAM as an application program, NJE can share the local communication facilities and communication link with other applications. At the same time, the synchronous data link control (SDLC) protocols and links used by the Network Control Program allow for full-duplex operation of the links.

And, finally, the Advanced Communications Function allows logical connections (sessions) to be established between processors that are not directly connected to one another. Since the routing of the data is passed only through the communications controller at the intermediate node, not only is there no requirement for spooling at the intermediate node, but the intermediate processor is not even aware of the fact that the job or data set has passed through the communications controller.

Additional uses of job networking

By combining the capabilities of job networking with other facilities of HASP, ASP, JES2, and VM/370, end-use functions can be accomplished other than those already described.

bulk data transfer

It is possible to transfer data sets from one NJE node to another using the job networking facility by first running a utility on the node from which the data set is to be transferred. This utility generates a job on the local node (through normal job entry facilities) which includes the data set to be transferred as an "in-stream data set" (i.e., a data set included with the job input). The job is then transferred via normal job networking facilities to the node to which the data set is to be transferred. When run, this job invokes a second utility that builds the duplicate data set from the "in-stream" data set included with the job.

The technique requires that the data set to be transferred be spooled at both the origin and destination nodes. This characteristic is not critical with relatively short data sets, but with large data sets, the disadvantage of having to spool the data sets these two times may be intolerable. At the same time, this method of transferring data sets is advantageous because it is totally integrated into the normal operations of the involved nodes. No synchronization is required to get the appropriate volumes mounted at both the transmitting and receiving nodes at the same time.

Although there are a number of utilities available to transfer data sets, there are no standards in this area, and incompatibility between various implementations of the utilities that have been implemented is common.

It is more efficient to transfer large data sets directly between applications running on the involved systems. Synchronization becomes a problem at this point, however, because the applications at both ends must be aware of the requirements to transfer data sets and know with which other application to establish a "session."

Even here, however, job networking can help. By transmitting the application as a job, the information concerning the type of data set and the name of the application that is sending it can be included as data with the job. Job management can even be used to get the appropriate devices mounted and allocated.

spool unloading and backup

There are times when the spool disk used by the job entry subsystem becomes excessively full. When this happens, it would be desirable to be able to temporarily unload certain jobs and/or system output data sets to another volume until a more opportune time. At such times, with job networking, selected jobs and system output data sets can be routed by the operator to other nodes in the job network where they can be held and then routed back at a later time.

In the case of JES2, the backup node can actually be another JES2 running as a secondary subsystem in the same CPU but with a different set of spooling volumes. The sole purpose of this secondary subsystem may be to offer additional spool space whenever the normal spool space becomes saturated, or it may have other functions which it performs. This secondary subsystem can be started and stopped as it is needed.

To communicate with this secondary JES2, some form of "wrap-around" communication facility must be provided. This requirement can be satisfied by a channel-to-channel adapter connected to two different channels on the machine, a binary synchronous

line that has been connected between two ports on the communications controller, or two dial ports temporarily connected together.

development using VM/370

VM/370 is an excellent system for interactive program development. As a "batch" system, however, it does not perform as well as some of the other systems that are designed especially for that aspect of data processing.

With job networking, one can really have the best of both worlds. Program development can be done on a system running VM/370, and batch work (such as large assemblies, compilations, and production jobs) can be "sent" to a system doing batch work via the job network. At the completion of the batch run, all desired output can be returned to the VM/370 system for verification or printing.

Summary and concluding remarks

Job networking provides a number of functions that can assist an enterprise in combining multiple computing facilities into an integrated computer network. Data sets (and data bases) that were previously available only at one location in an organization can now be made available throughout the entire organization. This availability can be accomplished either by sending jobs to the location that maintains the data set or, if timeliness is not extremely critical, the data set can be periodically transmitted to all desired locations using job networking.

In a similar way, reports can be distributed throughout a large organization with multiple computing centers by transmitting the reports as system output data sets. Thus, large reports can be made available very quickly over a wide geographical area.

Job networking can also assist the transition to new applications and new operating systems by making those applications and operating systems available through the job network for selective testing. Licensed products can also be made available to a far greater number of users in this way.

And finally, a powerful development tool can be created by combining the VM/370 Networking PRPQ either with NJE for JES2 or with one of the other NJI components. Development and testing can be done on the VM/370 system while production and batch compilation work is done on the batch system.

Combining separate computer facilities into an integrated job network cannot always be done without a great deal of effort on the part of the involved centers, however. Quite often the computer

facilities have grown autonomously for many years, and, during that time, many incompatibilities have developed between the centers.

For instance, the naming conventions used in different centers are not always in agreement. Names of volumes, data sets, devices, cataloged procedures, and applications are sometimes strange to the programmer attempting to prepare his job to run on a different system. Even the control cards themselves are different in the various types of job entry subsystems.

Also, accounting numbers and procedures usually vary from installation to installation. These incompatibilities were not critical when they were operating as separate centers, but, when they start to share work, they become significant obstacles that must be resolved before a substantial amount of work can be transported from one installation to another.

It can be argued that these incompatibilities will prevent widespread use of job networking. That does not seem to be the case, however, as the use of NJE and NJI has been increasing rapidly. For example, IBM's internal use of these programs now involves over 200 nodes. The benefits to be derived from such a capability appear to more than compensate for any difficulties experienced in learning to use it.

CITED REFERENCES

1. *Introduction to Advanced Communications Function, Multiple System Data Communication Networks*, GC30-3033, IBM Corporation, Data Processing Division, White Plains, NY 10604.
2. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15,** No. 1, 4–23 (1976).
3. *Network Job Entry Facility for JES2, General Information*, GC23-0010, IBM Corporation, Data Processing Division, White Plains, NY 10604.
4. *Program Product Design Objectives, Network Job Entry Facility for JES2*, GC23-0009, IBM Corporation, Data Processing Division, White Plains, NY 10604.
5. A. L. Scherr, "Functional structure of IBM virtual storage operating systems, Part II: OS/VS2-2 concepts and philosophies," *IBM Systems Journal* **12,** No. 4, 382–400 (1973).
6. *Network Job Interface, General Information: PRPQ P09007 (VM/370 Networking Prog 5799-ATA). PRPQ P09008 (HASP Networking Prog 5799-ATB), PRPQ P09009 (HASP Networking Prog 5799-ATC).* GH20-1941, IBM Corporation, Data Processing Division, White Plains, NY 10604. These PRPQ versions stem from custom work done in response to user's requests for specialized features.
7. *HASP Networking PRPQ P09009 Prog 5799-ATC, Specifications*, GH20-4526, IBM Corporation, Data Processing Division, White Plains, NY 10604.
8. *ASP Networking PRPQ P09008 Prog 5799-ATB, Specifications*, GH20-4525, IBM Corporation, Data Processing Division, White Plains, NY 10604.
9. *VM/370 Networking PRPQ P09007 Prog 5799-ATA, Specifications*, GH20-4524, IBM Corporation, Data Processing Division, White Plains, NY 10604.
10. *Introduction to JES3*, GC28-0607, IBM Corporation, Data Processing Division, White Plains, NY 10604.

Reprint Order No. G321-5071.