Computing system networks hold promise of increasing system efficiency through the sharing of resources, programs, and files.

Required is a protocol that performs for the network a function analogous to that performed by control blocks for operating systems.

Described are basic message-handling concepts and a protocol that are compatible with a broad range of network designs.

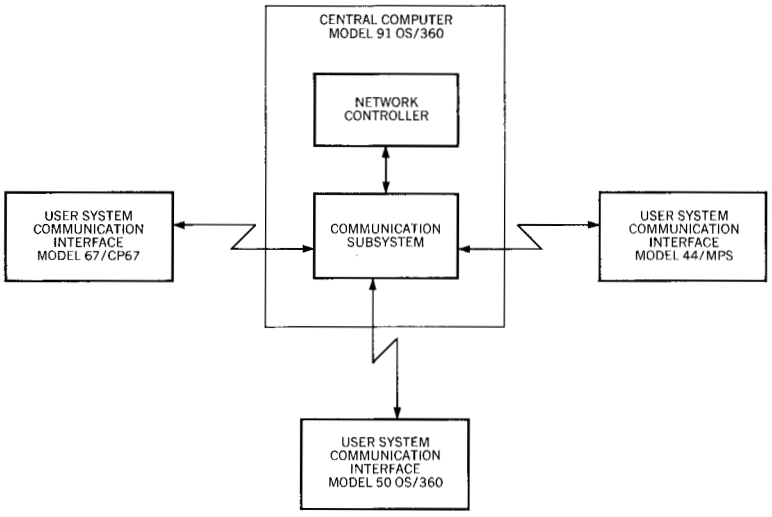
Protocol for a computer network

by D. B. McKay and D. P. Karp

In 1968, proof of feasibility was the motivation for the computer network project that was conceived and implemented at the IBM Thomas J. Watson Research Center, and in which the network protocol discussed in this paper was tested. At the time of the project's commencement, there were no operational computer networks. The Advanced Research Project Agency network[1] was also in the early design phase. Therefore, the IBM experimental network had the potential for making substantial contributions to a new field of computer application. Having no precedents and no other networks with which to coordinate, research was undertaken on a broad spectrum of network possibilities.

The one hypothesis of our network project was that the implementation take advantage of existing IBM hardware and operating systems. This was not viewed as a limitation, but as an opportunity to actually implement network programming concepts of general applicability rather than propose hypothetical designs that could be tested in only the most limited way. Thus, drawing upon the experience of communication networks, the classical star configuration was implemented, which involved principles that are applicable to distributed networks. In operation, these configurations embodied the following four models of System/360: Model 91 with OS/360, Model 67 with CP67, Model 44 with MPS, and Model 50 with OS/360. Systems were located in Yorktown Heights, New York, and Boulder, Colorado. Based

Figure 1 Integrated computer network in star configuration



on experience gained from the operation of the network, new insight into the purpose and problems of computer networks have led us to embark on a new study which is discussed at the conclusion of this paper.

Although details of the implementation have been given to lend concreteness to the concepts, the scope of this paper is to describe the fundamental concepts and definitions of the protocol used in the networks. It should be noted that the protocol is not restricted to any physical configuration.

the network

The objective of the network is to treat all systems involved as parts of a single, large multiprocessor system. Figure 1 is a simplified diagram of the network as it was organized in a star configuration. Everything that is said about the star is equally true of the distributed network except that in the distributed network each user system has the functions that are provided by the center of the star. The network consists of the following three basic components:

- Network controllers
- Communication subsystem
- User system communication interface

The *network protocol*, which is the main subject of this paper, is an established sequence of programming events for controlling communications among the network users in a well-defined, concise manner. The network accepts user command-language programs that state explicitly what is to be accomplished. The

network controller interprets these programs and issues the appropriate commands to manage the transfer of jobs and data through the network and to allow asynchronous execution of each user's network job. A *network job* for this system is a user-specified ordering of network commands on user systems that reference files and jobs located on various systems throughout the network. The network controller tracks the status of all user systems in the network and records such items as user system up, available for work, down temporarily, or down permanently. The network controller also manages network job queues to facilitate the simultaneous execution of user batch jobs. Executing jobs may also be network jobs initiated to transfer a file or program.

The functions of the *user system communication interface* are defined by the commands and protocol to be discussed. In general, it provides communication ability between the user system and the central system via a line-handling capability that adheres to the protocol of the communication subsystem. The user communication interface executes instructions sent to it by the network controller for the execution of network jobs. In performing these functions, the interface does two basic things—it loads jobs into the job stream and sends and receives files. Of course, the interface must allow the user to enter network command-language programs and transmit them to the network controller.

The *communication subsystem* is a store-and-forward digital switch that acts as a switching central for all messages traversing the network. The communication subsystem provides store-and-forward switching and temporary storage for messages destined for unavailable user systems. The subsystem delivers these messages when the user system becomes available. The communication subsystem manages all messages in the network queue and transmits them to the proper destination, thus acting as a concentrator-multiplexor and traffic director for the network. The network controller is treated as another user by the communication subsystem.

We now discuss fundamental concepts of messages as they relate to the network just described.

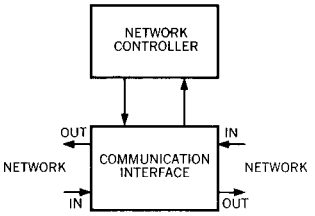
Fundamental concepts

It is necessary to delineate computer network control functions such as switching control, buffer control, message control, and operations control when establishing a network operational protocol. The operational control function is further complicated because the user can program the network as though it were a

single system. Consequently, the protocol is broken down into detailed functions to cope with events such as error recovery and handling techniques. Performing these functions are the two basic realms of control—network and communication. *Network control* is a higher-level function that recognizes and controls all aspects of network jobs and the execution of job steps in the network machine. *Communication control* (performed by the communication subsystem) provides fast switching and buffering services for all messages moving between user systems without the intervention of the network controller.

Figure 2 illustrates the relationships of these two realms of control for messages travelling into, out of, and directly through a communication subsystem, including control of messages that must be acted on by the network controller. Messages requiring the network controller are discussed later in this paper. Network and communication functions can exist on any system and operate in any physical configuration provided that the control information in the messages reflects the configuration so that proper operation is maintained. There is no reference to physical configuration in this paper beyond the experimental configuration mentioned in the introduction because of the flexible nature of the protocol and its adaptability to any configuration. For example, in the case of a distributed network, the communication subsystem passes messages directly to the next station without any network controller overhead. The network controller comes into play only at the final destination and, although present, has no interaction at intermediate stations.

Figure 2 Functions of the communication interface

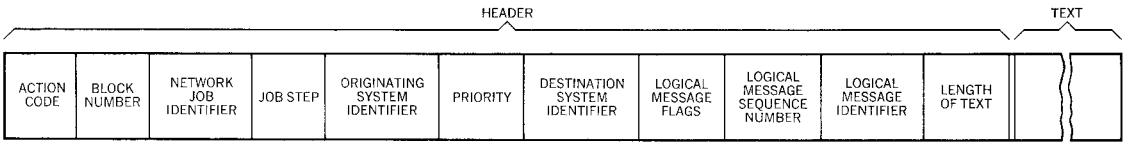


message concepts

Before discussing the network operation and control protocol, basic message content and concepts must be defined. A *transmission block*, as indicated in Figure 3, is a physical message entity that consists of header and text. A logical message consists of many transmission blocks. The header uses a set of operational functions to provide all the information necessary to maintain control of message transmission from one user system to another.

An *action code* specifies the immediate destination of each transmitted block. The appended text may be transmitted directly to the user, who is identified as the destination system in the header, or sent to the network controller or used by the communication subsystem. Conflicting information between the action code field and any other field in the header causes an error message to be returned to the originating station. The action code serves a similar function in the receiving system, indicating to the user communication interface whether the data block is destined for a user routine or contains control information for the user system communication interface, which coordinates directly with the local user operating system.

Figure 3 Transmission block



A transmission *block number* is a serial number inserted by the original transmitting station into each block being transmitted within the network. As the block flows through the network, each successive station inserts its own number into the block, overlaying the previous station's number. The purpose of the transmission block number is to provide that no messages are unaccounted for in the physical communication process.

The *network job identifier* is a field that associates transmission blocks with the network job to which they belong. The identifier is assigned to the network job and to each associated transmission block by the user system or by the network controller. A unique name for each job within the network is formed by concatenating the name of the user system that is originating the network job with a number generated by the originating user system.

A *job step* marker in the header uniquely identifies a job step within a network job. The network controller assigns this name since it maintains control of all network jobs.

An *originating system identifier* is a unique name associated with each user system for routing a block of data from one user system to another. The identifier is assigned arbitrarily but uniquely by the network management group at the time the user system is accepted as a network participant. The station originating a block of data places the assigned identification into this field.

The message *priority* field indicates transmission priority (which is not the same as processing priority) by block within the queue for a particular user system.

The *destination system identifier* is similar to the originating system identifier except that the identification inserted is that of the system for which the transmission block is destined.

Logical message flags denote the first and last blocks of a message; all intermediate blocks are noted by the absence of such flags. The flag field in conjunction with the logical message sequence number enables the user to determine blocks missing

from a message, and provide an indentifier that can be used to recover missing blocks. When the first and last indicator flags are turned on in a single transmission block, the complete message is contained within that block.

The *logical message sequence number* field numbers sequentially the blocks within a message. The first block is denoted by the *logical message flags* and contains the lowest number assigned (not necessarily 1) within a message and the last block contains the highest number. Unlike the transmission block number, the logical message sequence number remains intact throughout the journey of the block through the network. Together with the logical message flags, it is used for error detection and recovery.

A *logical message identifier* is provided and used to reassemble a message at the user destination, because all communication lines in the network can be multiplexed, thereby causing transmission blocks within a message to be interleaved with blocks from other messages. Therefore, each block within a message contains this unique message identifier. In the simple case where the message contains only one block, the logical message identifier performs no function. When multiple blocks comprise a message, the logical message identifier enables the user to reassemble the message. There can be any number of physical message blocks associated with a given logical message. It is important that the network controller assign numbers to this field for all messages it augments, and that the logical message identifier be used in the messages generated by the user communication interface in response to network controller commands.

Table 1 Logical message structuring

Network controller

NJID 1
- Stepname 1
  - LMID 1
    - LMSN1
    - LMSN2
    - .
    - .
    - .
    - LMSN *k*
  - LMID 2
    - .
    - .
    - .
  - LMID 1
- Stepname 2
  - .
  - .
  - .
- Stepname *m*

NJID 2
- .
- .
- .

NJID *n*

The *length of text* is a field that contains a binary number equal to the number of characters in the text portion of the transmission block. Although there are other means available to obtain this number, it is included in the header as a redundancy check.

logical message structuring

The network controller maintains control of every user job submitted by using the network job identifier (NJID). Beginning with the NJIDs, hierarchical structure is built up based on the message configuration. Any message pertaining to any step in a network job can be tracked and retransmitted if necessary. The message hierarchy provides a mapping of the logical structure of each network job into its appropriate message configuration. The network controller in combination with the communication subsystem and the hierarchical message configuration make it possible to record and reflect the global status of the network to any desired level of detail. In terms of the message concepts just described, the communication subsystem has the ability to keep track of all messages in the network by a combination of network job identifier, logical message identifier, and logical message sequence number as shown in Table 1.

Network protocol

So far we have been concerned with the intervals of message structure and the logical relationships of messages to the computer network. We now discuss rules of procedure for interactions among the user systems of the network. Collectively, these procedures are the network protocol. Thus a protocol bears somewhat the same relationship to the network as a control block bears to an operating system. The protocol for operating a network system has different levels of control. For example, the communication subsystem exercises control on the communication link between any pair of user systems, and the network controller maintains control at the network job level. Further, the functions that each of these units perform are combined to handle special control cases. These complementary control functions are discussed in greater detail as they arise naturally out of the details of the protocol.

initialization

The first network procedure is a series of initialization messages sent from one user system to another before any message transmission takes place. Initialization messages are sent between each affected user system with positive acknowledgments received to complete the initial hand shaking. At any point during the transmission of initialization messages an error can occur and be detected by a negative acknowledgement. The message in error is retransmitted several times. If the error persists, the line is timed out and retried later, on the assumption that the line is temporarily noisy and requires time to quiesce.

When a user system receives an initialization message, it responds in one of the following ways, depending on the status of the user system:

- Ready to receive and transmit.
- Temporarily, cannot receive actual data transmissions, but can receive special control messages. (This option allows a user system to selectively process network jobs as user facilities become available.)
- Physically connected, but cannot accept traffic from the network.
- Unable to receive new network job requests, but handling traffic for jobs in progress. (The user system may have several jobs in progress that are transmitting and receiving messages. This acknowledgment allows the user system to continue normal processing of jobs in progress.)

clearing up the network

Thus the communication interface of each user system can selectively demultiplex itself to permit its handling of only one logical message, or suspend message transmission for any task that is temporarily deactivated. The destination system can

selectively halt all messages that have a given network job identifier or selectively halt logical messages within a net job. An adjacent system may continue accepting messages until its buffers are filled to a specified operational threshold limit that protects the network from coming to a standstill because of saturation. Thereafter, selective halts are issued to the systems sending to it. Message blocks of one logical message can be stored in distributed segments throughout the network.

The same selective halting mechanism can be applied in reverse through a *resume message*, which can apply to an entire set of messages for a network job or selective logical messages within a job. The reinitiation of a transmission takes place between any two user systems that wish to allow more message blocks to be transmitted. The destination system must resume on a particular logical message to allow the message to reach its final destination and complete transmission through the network. The logical message identifier of the message header enables the communication subsystem and the network controller to cooperate in controlling and clearing up network operation. Not only does this cooperation between logical levels reduce a duplication of effort, but it also enables the control to become realistic and practical.

The inability to transfer communication and control information between the two functions results in a less effective overall network control. For example, if a message consists of many blocks and an unrecoverable transmission error occurs, the communication subsystem notifies the network controller of the error occurence, then the network controller submits purge messages to the communication subsystem for those particular logical message blocks. This mechanism allows a general clearing up and management of all message transmissions. In the actual operation of the network, the user is notified of this condition so that he may resubmit his network job.

Another emergency condition occurs when a receiving user system goes down, and there are a number of network jobs involved with that user system. If the user system remains down for an extended period of time, and the communication subsystem buffer resources are filled to the threshold limit, it may be necessary to purge message blocks addressed to the down user system. The communication subsystem notifies the network controller of the user system's being down, and the network controller issues purge commands to the communication subsystem for all pending messages of those network jobs involved with the down user system. In our present implementation, the communication subsystem uses disk storage as a logical extension of main storage for message buffering. In this operation, the freeing of real main storage buffers becomes a simple matter of

moving messages to disk for later retrieval. In some instances of transmission, a file may be stored in segments at several locations until the receiving system can receive it. Network buffer resources are treated as a logically simple entity, which may, in fact, be physically distributed.

When the user system comes back on the air, the involved user network job is restarted by issuing resume transmit commands to the communication subsystem. If the user is in an interactive mode of network control, he is notified of the problem and status of his file transmission, and he can reinstate his command then or at a later time. The batch network job is restarted at a point where no unnecessary retransmission need occur.

It has not been determined how long files should reside in a store-and-forward location before being purged from the network. If a backing storage device is available to network operation, the file can remain for a longer time but still not indefinitely.

network controller batch protocol

The batch file transmission protocol of the network controller is primarily concerned with the control and transfer of user files for storage, temporary use at a remote system, and execution. *User* is defined as the user system that is using or has the intention of using files or data in the current transaction. The user of a given file can change identity from transaction to transaction; the concept of user does not imply ownership. The commands and status messages that pertain to the second-level logic of the network controller are transmitted by the sending system and interpreted by the receiving system. All initiations of file transfers are directed from the command-sending user to the network controller.

These commands interrogate the file-sending system to determine whether a file is resident at that system. If it is not catalogued at the file-sending system, the user must provide the necessary information to locate a file. This information consists of such physical attributes as file identification and volume serial number. A negative acknowledgment of this command results in the termination of a net job step with the reason for termination returned to the sender of the original request. When a positive acknowledgment is received by the network controller, it has the following two options available. After determining the amount of unused buffer space in the communication subsystem—and based on the size of the file to be transferred—the network controller decides whether to: (1) send the data set immediately or (2) wait for an acknowledgment of the receive message.

If the network controller decides to move the file regardless of the state of the receiving system, the controller issues a send

command to the sender and a receive command to the receiver simultaneously. A negative response to the receive command is taken as a definite refusal by the receiving system to accept the data transmission. This may result from insufficient resources to handle the job. If the file is transmitted from the sending system and becomes resident in the network storage facilities, the user (receiver) is notified of its exact location so that he may move it from that point at a later time. If the network controller chooses the second option, the file remains resident in the originating (sending) system.

A positive acknowledgment allows the file to continue its normal flow through the network. Queuing in the communication subsystem is always done so that receive commands are transmitted before the actual data file. Receive commands specify the action to be taken with each file. Such actions include loading a file directly into the job stream (which assumes that appropriate JCL commands are included in the text of the files), or cataloguing a file at a remote system, or storing a file for use. All network files are catalogued with a unique name that includes the following user specifications:

- Data name – unique among all user files
- Identification – unique among all users of the user system
- System identification – unique among all user systems in the network

A receive command may also contain special instructions to print or punch a file.

When a sending and a receiving system complete a file transfer, they each send status messages back to the network controller that indicate the completed action. These status messages enable the network controller to record user network job steps and their progress through the network, and they play an important part in providing proper checkpoint restart for the network.

Files routed specifically for execution require a third status message from the receiving (user) system that indicates the time and the manner in which the job has been executed. This status message also contains the appropriate accounting information to allow dynamic updating of network user and system accounting information. It is not clear at this time what should be accounted for in the network, but this is an area of prime concern to operational networks.

An error in the second logic level can occur during the file transmission. There may be an error moving files from devices into the line buffers or reading from the line buffers. When this occurs, the operating system must pass this information to the

network controller, which terminates the task involved in this job step and purges all the network buffers containing blocks of this message transmission.

When the network controller receives the file error command, a release command is immediately sent to all the network tasks supporting that job step. This action causes the user systems to end all pending tasks associated with that network job step. In addition, a purge command for that job step is sent to the communication subsystem to purge the message from its buffers. If there is more than one communication subsystem involved, the purge message is passed to all other such subsystems. This is another example of the communication subsystem and the network controller combining functional capabilities to provide effective management of network traffic. The mapping of a message into the job step allows the network controller to selectively choose all messages it wishes to purge.

network controller interactive protocol

For interactive operation, the network controller requires a protocol that differs somewhat from that required for batch operation. Standard message types are provided for interactive use for proper message recognition among user systems. Terminal-type traffic is transmitted through the network via the normal user system interfaces. Control information transmitted by a terminal to the user's operating system is incorporated in the network protocol by the user communication interface.

The interactive user requests a direct connection to the remote system through the network controller, which, in turn, notifies the remote system of the user request and establishes a direct link between the two systems. The network controller then monitors the conversation but is not directly involved with the messages. Conversational messages are interchanged via the communication subsystem with no interaction with the network controller. In the event that one of the systems goes down, breaking the logical link, the network controller notifies the other system to terminate the waiting task. In most cases, a down user system is isolated from the second user system by other stations, and the network controller is a convenient way of notifying other user systems of the loss of service.

When the user's connection is established, the following three types of messages, which are included in the protocol, may be generated, and they are identified by the action code field in the header:

- Response request, with or without text included in the message
- Text message, which may be the response to a request or data to be printed at the user terminal

- Interruption, which signals that the user is stopping a task or wishes to talk directly to the opposite operating system

Regardless of network conditions, there are always sufficient buffers to receive an interruption message and terminate or purge an existing task together with the associated operation it may be supporting.

Concluding remarks

The primary intent of the protocol presented in this paper is to facilitate and control the transfer of data through a network of computers. Although the protocol effectively deals with this problem, it only begins to solve the problems of a resource-sharing and distributed-data capability that networks should provide. The problems of managing data on a single system are difficult, but a multisystem data-management capability is probably an order of magnitude more difficult.

There is a consensus among current network users that networks can provide more effective utilization of computer resources than is presently the case. This capability will only be realized when data and system functions can be employed more effectively and easily by remote systems.

There is still a large set of problems that remain unsolved, but which are being addressed in IBM Research. For example, how completely can disparate data files and system files be shared among different systems? What degree of control can an overall network control system impose on a local systems controller? The problem of a high-level user's interface language and the degree of machine independence that can be achieved with it are also being studied.

The high-level user language problem is that of trying to achieve a multimachine environment that is transparent to the users. The ultimate objective is to make networks as easy to use as possible. With economy and ease of use, applications and the number of users can increase to the point where networks can sustain themselves.

CITED REFERENCE

1. L. G. Roberts and B. D. Wessler, "Computer network development to achieve resource sharing," *AFIPS Conference Proceedings; Spring Joint Computer Conference*, **36**, 543–549 (1970).