

UC Irvine

ICS Technical Reports

Title

Interprocess communication in a dataflow system

Permalink

<https://escholarship.org/uc/item/0hg7p4wc>

Author

Beaty, Claudia Ann

Publication Date

1979

Peer reviewed

INTERPROCESS COMMUNICATION
IN A DATAFLOW SYSTEM*

by

Claudia Ann Beaty

DEPARTMENT OF
INFORMATION AND COMPUTER SCIENCE
UNIVERSITY OF CALIFORNIA, IRVINE
IRVINE, CALIFORNIA 92664

INTERPROCESS COMMUNICATION
IN A DATAFLOW SYSTEM*

by

Claudia Ann Beaty

Technical Report #137

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

June 1979

*Master of Science Thesis - UCI Dataflow Architecture Project.

ABSTRACT OF THE THESIS

Interprocess Communication
in a Dataflow System

by

Claudia Ann Beaty

Master of Science
in Information and Computer Science

University of California, Irvine, 1979

Professor Kim P. Gostelow, Chair

This thesis investigates interprocess communication for the dataflow system being developed as part of the UCI Architecture Project. It includes a survey of existing interprocess communication mechanisms. Two communication facilities developed for the dataflow system are presented along with a comparative analysis of the facilities. One of the communication facilities not only provides a means of logical communication but also provides a means of solving a problem concerning the destruction of dataflow managers.

INTRODUCTION

The objective of this thesis is to develop an inter-process communication facility for a dataflow system, in particular, the system being developed as part of the UCI Dataflow Architecture Project [AGP78]. There were several motivations for the development of a communication facility including the potential for logical communication. Logical communication allows a process to communicate with dynamically specified processes, adding flexibility to a computer system. This flexibility and its advantages are evident, for example, in the UNIX operating system [RT74] where a process is able to communicate with other processes without knowing which processes they are or what their structures are.

Another motivation for the development of a communication facility was the potential for solving a problem concerning the destruction of dataflow managers. This problem and its solution are discussed in Appendix A.

The dataflow model and the high-level language Id (for Irvine Dataflow) presented in [AGP78] were utilized for the development of a communication facility. This development was initiated by a review of the communication mechanisms existing in conventional systems. An evaluation of these mechanisms is given in Chapter 1. This evaluation led to the development of the Message Passing Facility described in Chapter 2. The basic communication mechanisms already

existing in dataflow led to the development of the Logical Assignment Facility described in Chapter 3. A comparison of the Message Passing and Logical Assignment communication facilities resulted in the conclusions that the Logical Assignment Facility is more flexible, more efficient, and easier to utilize. The comparison and conclusions are included in Chapter 4. The remainder of this introduction is directed at explaining the basic principles of dataflow and making the reader more familiar with some of the concepts of Id.

A dataflow system is a computer system with an architectural basis that differs from that of conventional von Neumann machines. Conventional systems, used in a distributed fashion, contain two features considered to be causes of difficulty - sequential control and the memory cell. This contention and the arguments that support it are given in [Dennis73, GIMT74, Backus78, AGP78]. The rejection of sequential control and the memory cell is embodied in two basic principles of dataflow. These principles are [AGP78]:

1. An operation executes when and only when all of the operand values needed for the operation are available, and
2. An operation is purely functional and produces no side effects as the result of its execution.

Several of the concepts described in [AGP78] are of particular interest with regard to interprocess communication and are described here. Structure, procedure definition, manager definition, and manager instances are primitive types of Id operands. A structure value is either the empty structure or a set of ordered pairs <selector: value> where "selector" an integer or a string value and "value" is any Id value.

A procedure definition value is a model from which many instances of the procedure can be created. The created instance of a procedure definition exists only during a single procedure application. A procedure may be applied to an argument by supplying the procedure definition and the argument to the apply primitive. The result of the apply is the result produced by the procedure. If more than one argument is required by the procedure, a structure parameter that contains all of the arguments will be supplied to the apply primitive.

Similarly, a manager definition value is a model from which many instances of a manager can be created. However, a created instance of a manager is not necessarily destroyed each time it is used. A manager instance may be created by supplying the manager definition and initialization parameters called creation-time parameters to the create primitive. An instance of a manager may be used by supplying an input value to the use primitive. The input value is

sent to the manager and the result produced by the manager is returned as the result of the use operation. Many users may use a particular instance of a manager, each supplying an argument that is processed in a non-deterministic basis. The input argument is sent to an entry port of a manager. Since a manager may have multiple entry ports, the entry port to be used is specified in the use expression.

With these basic concepts of the dataflow language, it is possible to define a "process." In conventional systems the notion of a process is usually described in terms that relate to sequential control or memory - features that are rejected by dataflow. [Dennis, Van Horn66] define a process as "a locus of control within an instruction sequence." [Brinch Hansen70] defines a process as "the execution of one or more interruptable programs in a given storage area." There is no equivalent notion of a process in a dataflow system. However, a process can be thought of as an entity that has the need to communicate with other entities. In a dataflow system, a process most naturally corresponds to an instance of a dataflow manager. Therefore, a process is defined as an instance of a dataflow manager.

Chapter 1

A SURVEY OF INTERPROCESS COMMUNICATION MECHANISMS

The nonexistence of previous work in the area of communication facilities for dataflow systems led to an evaluation of communication mechanisms in conventional systems. In conventional systems mechanisms for interprocess communication have taken two forms - shared data, and message passing. Shared data schemes allow processes to modify and acquire information through synchronization controls. Since the use of shared data is incompatible with the principles of dataflow, the second approach, message passing, was evaluated as a basis for a communication facility.

In a message passing system, a channel or a port is a path of communication between processes. This mechanism includes a set of operations on a channel that allows information to be sent from one process to another process.

[Hunt78] has included in a paper, "Messages in Typed Languages," a critique of the various message passing schemes described in the literature. Several features of message passing systems are notable:

Channels may be associated with processes or with values. If a channel is associated with a process the path of communication is fixed. A channel must exist for every pair of processes that wish to communicate. However, if a

channel is associated with a value, the path of communication is not fixed. Values may be passed from one process to another, affecting a change in the path of communication from one set of processes to another set of processes. That is, identifying channels by values provides a means of logical communication.

Message passing schemes differ in the number of senders and in the number of receivers possible in the system. Some facilities allow no more than one sender and one receiver per channel [Hoare78]. Walden's system [Walden72] allows multiple senders and no more than one receiver per channel, even though the potential for multiple receivers exists. The interpretation of multiple senders and multiple receivers will be as given by [Hunt78]: Sending a message on a channel with several receivers ensures that the message will reach exactly one of the receivers, while receiving a message from a channel with several senders causes a message to be received from exactly one of the senders. A process cannot specify which of the several senders or receivers it will communicate with. The nondeterminacy provided by multiple senders and multiple receivers is a useful feature. [Hunt78] uses this feature to provide scheduling for non-preemptible resources.

A connection may exist over a series of messages or it may exist for exactly one message transfer. In the first case, there is less overhead involved in prolonged

communication between two processes. In a message passing facility there are several advantages to the second which include the following. There is no need to set up and break connections. The receiving process can stop the flow of messages or information after each message transfer. Errors have a less serious effect. And it is easier to move channels from one process to another process.

Another feature of message passing facilities that is important concerns the generality of the system. A more general approach provides communication between remote as well as local processes.

There are other features of message passing facilities to deal with problems that do not arise in a dataflow system. For example, there is nothing in a dataflow system that corresponds to an "inactive" or "sleeping" process. Since dataflow processes are driven by data, there is no need to "activate" or "wake-up" processes that are to receive messages.

An important feature of a message passing facility is how provisions are made for buffering messages. If message buffering is provided by the communication facility, there exist problems concerning the length of the buffers. Message buffering is not provided by the Limp communication facility [Hunt78] so the length of buffers is not a problem. However, a process sending or receiving a message on a channel must be suspended until a partner requests the

complementary operation on the same channel. Dataflow rejects the use of the memory cell at the logical level. Although the problem of message buffering is not avoided at the physical level, it does not arise in the development of a communication facility for a dataflow system. It is instead a problem that is solved as a fundamental part of any dataflow system - regardless of the presence or absence of a message passing facility. Thus there is no need to consider the problem here.

In summary, interprocess communication facilities in conventional systems consist of shared data schemes and message passing schemes. Message passing facilities, those most compatible with the principles of dataflow, have certain desirable features including: the association of channels with values, the possibility of multiple senders and multiple receivers on a single channel, a connection that exists for one message transfer, and provisions for remote as well as local communication.

Chapter 2

THE MESSAGE PASSING COMMUNICATION FACILITY

BACKGROUND

Walden's system [Walden72] has several of the desirable features of message passing systems. Hence, Walden's message system was used as a basis for the development of a message passing communication facility.

In the Message Passing Facility described here, channels are used to transmit data between processes. The data transmitted on a channel, referred to as messages, may be ordered. Channels are associated with values, so channels may be moved from one process to another process. There is no limit on the number of senders or the number of receivers possible on a single channel. And there are provisions for remote as well as local communication.

A dataflow computer system consists of one or more dataflow machines. Each dataflow machine will be referred to as a site. Two processes located in the same site are considered local processes. Two processes located in different sites are considered remote processes.

DESCRIPTION

The following is a description of a set of operations enabling interprocess communication in a dataflow system. A dataflow manager, referred to as an Interprocess Communication (IPC) manager, performs the following operations:

1. RECEIVE. This operation allows a process to receive a message from another process. The operation has the following parameters:
 - a. Channel Number. An identifying number associated with a particular channel.
 - b. Site Number. An identifying number associated with a particular site, the site where the operation will be performed.
 - c. Sequence Number (optional). A value indicating the ordering of the messages to be received.
2. SEND. This operation allows a process to send a message to another process. The operation has the following parameters:
 - a. Channel Number. An identifying number associated with a particular channel.
 - b. Site Number. An identifying number associated with a particular site, the site where the operation will be performed.
 - c. Message. The data to be transmitted.
 - d. Sequence Number (optional). A value indicating the ordering of the messages to be sent.
3. UNIQUE. This operation obtains a unique number from the communication facility.

A channel is a communication path between processes. Each channel is identified by a unique number which is associated with the channel. This number is provided by the communication facility via the UNIQUE operation.

Channels are used in transmitting messages from one process to another process as follows. Process A executes a RECEIVE on channel X with site number Y, and sequence number Z. Process B executes a SEND on channel X with site number Y, message M, and sequence number Z. The parameters arrive at the IPC manager located at site Y where the channel numbers and sequence numbers are matched, and the message M then moves from the IPC manager to Process A (Figure 1A through 1C).

The SEND operation is implemented as a procedure call with the parameters of the SEND passed as arguments to the procedure. The procedure creates a structured parameter P and executes a use of the local IPC manager. The IPC manager will return an acknowledgment which may be an error response. This acknowledgment will then be returned as the result of the SEND operation (Figure 2).

Similarly, the RECEIVE operation is implemented as a procedure call with the parameters of the RECEIVE passed as arguments to the procedure. The procedure creates a structured parameter P and executes a use of the local IPC manager. The local IPC manager will return the message that

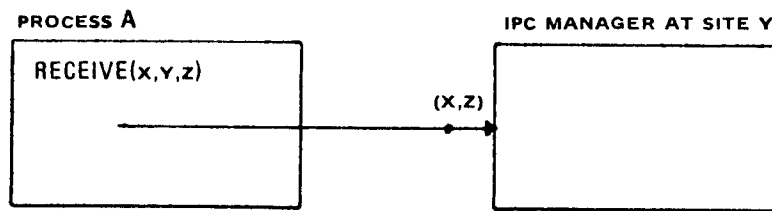


FIGURE 1A. PROCESS A EXECUTES A RECEIVE ON CHANNEL X WITH SITE NUMBER Y AND SEQUENCE NUMBER Z.

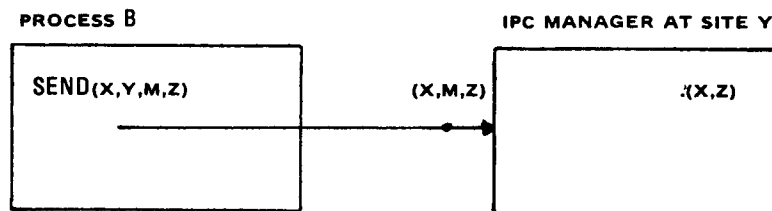


FIGURE 1B. PROCESS B EXECUTES A SEND ON CHANNEL X WITH SITE Y, MESSAGE M AND SEQUENCE NUMBER Z.

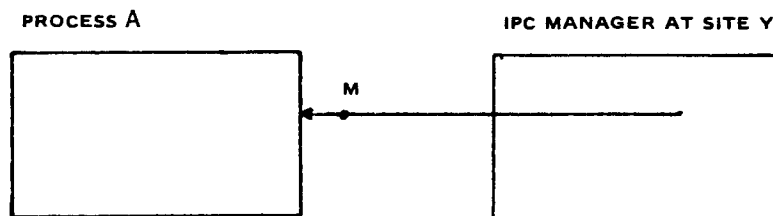


FIGURE 1C. THE CHANNEL NUMBERS AND SEQUENCE NUMBER ARE MATCHED AND THE MESSAGE NUMBERS MESSAGE M MOVES FROM THE IPC MANAGER TO PROCESS A.

Use of Channels in the Transmission of Messages

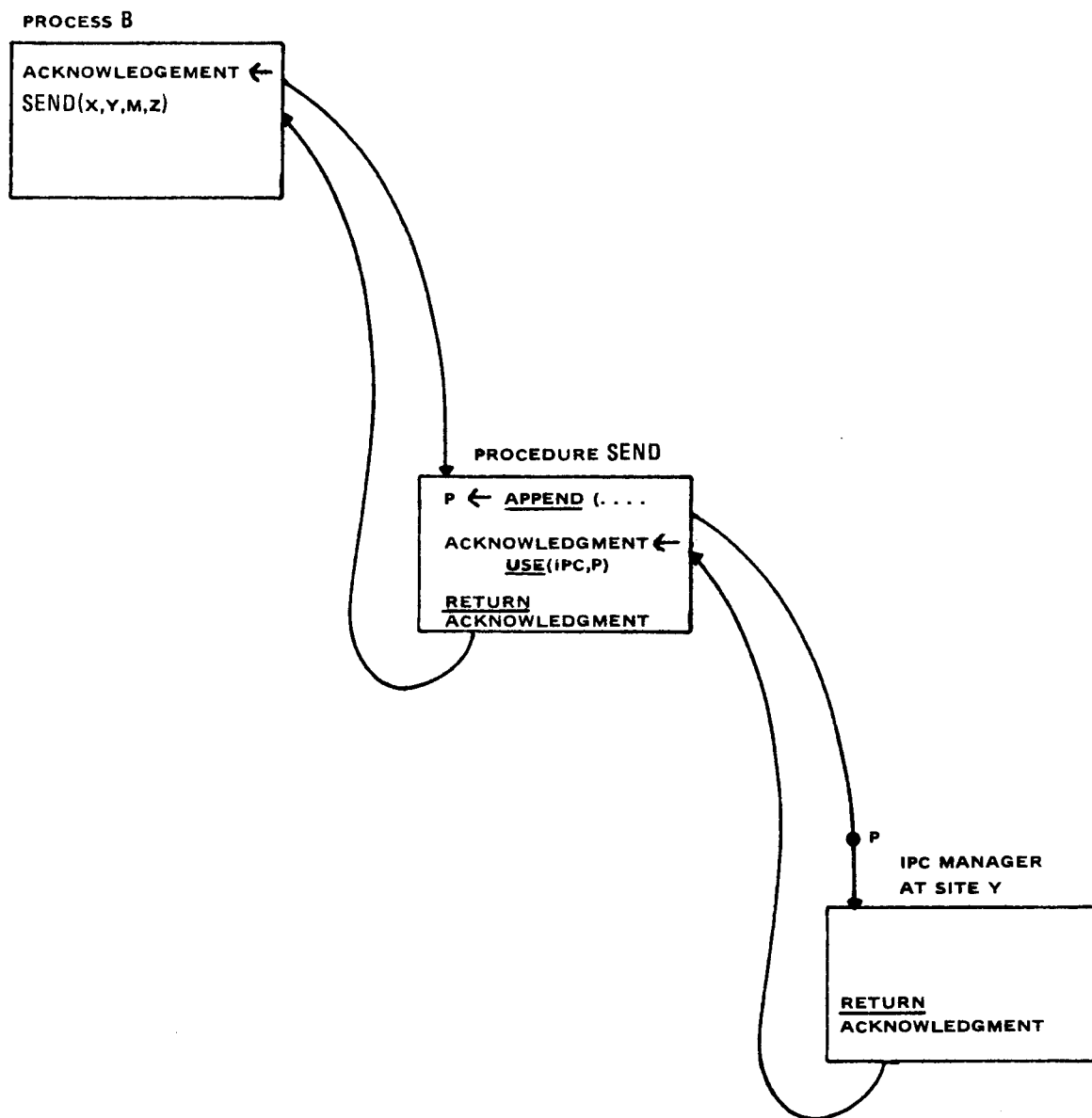


FIGURE 2. IMPLEMENTATION OF THE SEND OPERATION

was passed as parameter M in the complementary SEND operation. The message M will then be returned as the result of the RECEIVE operation.

In order to provide communication between remote processes, the operations are carried out by an IPC manager at each site within the computer system (Figure 3). Within each site there must exist information on other sites and the ability to transmit a SEND or RECEIVE request to other sites. How this is done is not important to this discussion. Therefore, for our purposes there exists a send expression which transmits a request to another site and returns the result of the request. The expression has two parameters: site number (the identifying number associated with the site where it is to be transmitted) and the request to be transmitted.

A SEND or RECEIVE request is initially sent to the IPC manager local to the process executing the request. The local IPC manager will check the site number to determine if the request is to be matched locally or if the request is to be sent to another site to be matched. The site where the SEND/RECEIVE pair is matched is called the rendezvous site. If the rendezvous site is a remote site the IPC manager will send the request to the remote site. The local IPC manager will then return the result of the send expression to the process executing the SEND or RECEIVE operation.

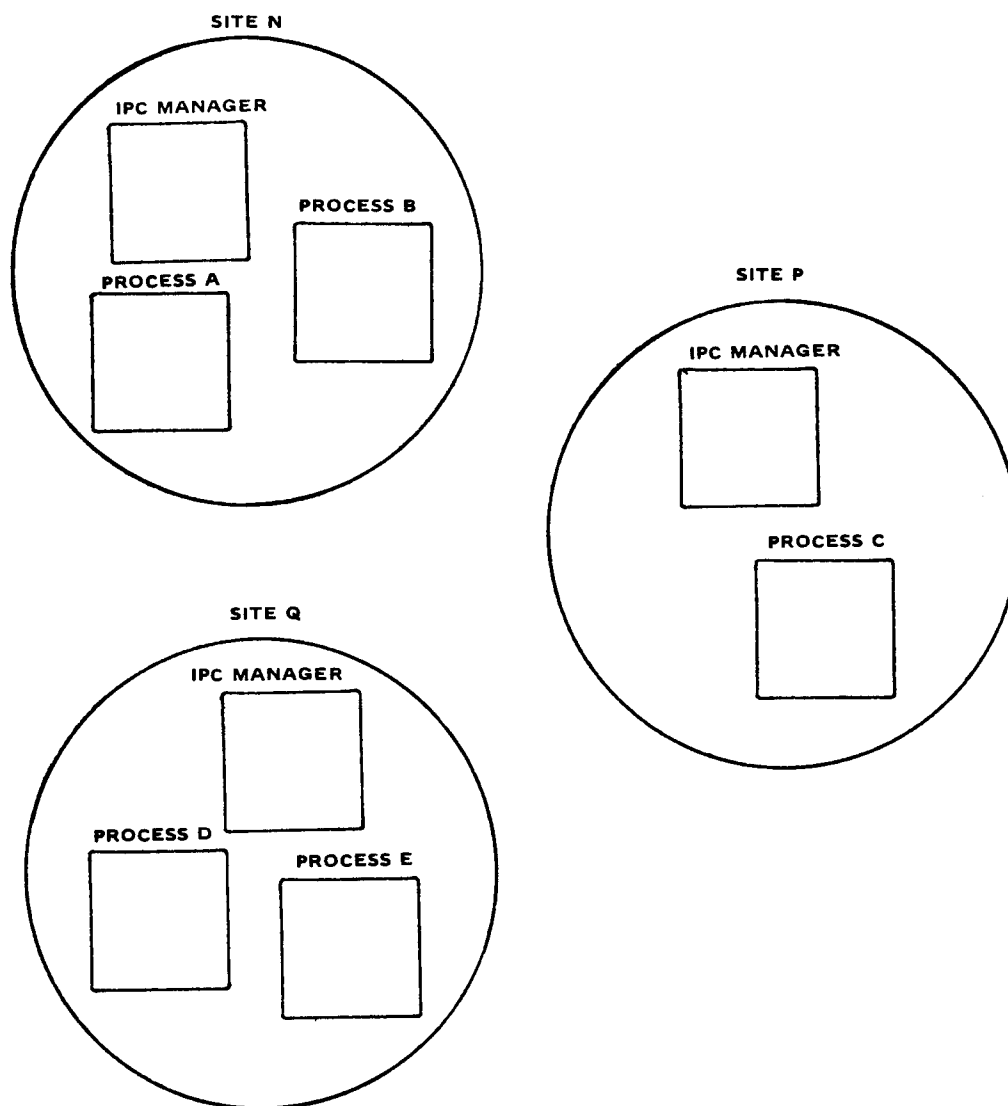


FIGURE 3. AN IPC MANAGER IS LOCATED AT EACH SITE WITHIN THE COMPUTER SYSTEM.

When remote processes communicate the rendezvous site is normally the site of the process executing the SEND. Under these circumstances, the SEND request will stay at the IPC manager local to the sending process until a matching RECEIVE request arrives. The RECEIVE request will first be sent to the IPC manager local to the receiving process. The IPC manager at this site will then send the RECEIVE request to the rendezvous site, where the SEND/RECEIVE match can occur and the message will be transmitted. With this convention it is possible to block message transmission from one site to another until a process is ready to receive the message.

ORDERING OF MESSAGES ON A CHANNEL

In a dataflow system messages and requests to receive messages may arrive at the rendezvous site in an indeterminate order. For example, this might result from an unraveled loop containing a SEND or a RECEIVE operation. Consequently, it may be necessary to order messages. Ordering is provided by the sequence number (an optional parameter), a value that allows two processes to synchronize the initial SEND and RECEIVE requests and to order the remaining messages on a particular channel.

If ordering is required, the user programmer must provide a sequence value as a parameter in the initial operation on a channel. The user programmer is then responsible for incrementing the sequence value and

providing the sequence parameter in communication operations on that channel.

If the sequence number S is provided as a parameter in a SEND or a RECEIVE operation, a message will be transmitted when the complementary RECEIVE or SEND is executed on the specified channel with sequence number S . For example, the result of a RECEIVE operation with a sequence number of S will be the message that was passed as a parameter in a SEND operation on the same channel with the same sequence number of S .

If the sequence number is not provided in a SEND or a RECEIVE operation, no ordering is provided by the facility. A message will be transmitted on the specified channel when a complementary RECEIVE or SEND is executed on that same channel.

MOVING A CHANNEL

An advantage of having channels associated with identifying numbers rather than processes is having the capability of moving a channel from one process to another process. This is easily accomplished with a receiving process. A process, P_1 , which has the information necessary to execute a RECEIVE on a channel may pass this information to another process, P_2 . The information that must be passed is the channel number, the site number, and, if ordering is required, the sequence number. This information can be passed either in a message or as creation-time parameters of

P_2 , a dataflow manager. The second process, P_2 , can then execute a RECEIVE on the moved channel and the sender never needs to know the channel has moved.

Moving a channel from a sending process, P_3 , to another process, P_4 , is easily accomplished if P_3 and P_4 are local processes. A process, P_3 , that has the information necessary to execute a SEND on a channel may pass this information to P_4 . The information that must be passed is the channel number, the site number, and if ordering is required, the sequence number. This information can be passed either in a message or as creation-time parameters of P_4 , a dataflow manager. The second process, P_4 , can then execute a SEND on the moved channel and the receiver never needs to know the channel has moved.

Moving a channel from a sending process, P_3 , to another process, P_4 , is slightly more complicated if P_3 and P_4 are remote processes (Figure 4). Earlier a convention was established for the location of the rendezvous site in remote communications. Normally, the rendezvous site would be the site of the sending process. So if a SEND process changes from P_3 to P_4 , the rendezvous site should change from the site of P_3 , say S_3 , to the site of P_4 , say S_4 . The receiving process will execute a RECEIVE specifying site S_3 and the RECEIVE request will be sent to site S_3 . P_4 will execute its first SEND specifying S_3 , the old rendezvous site (Figure 5). The SEND and RECEIVE will be matched at

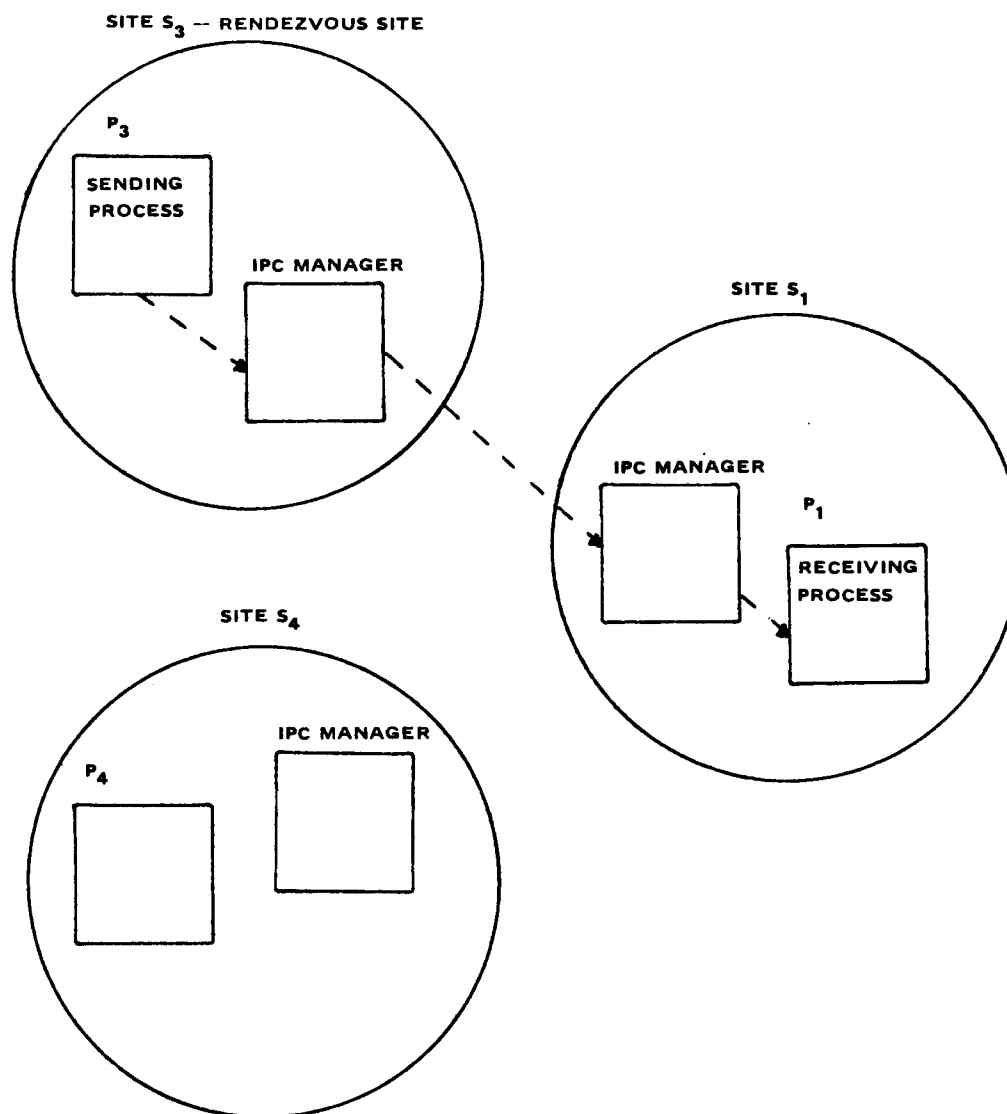


FIGURE 4. THE MESSAGE TRANSMISSION FROM P_3 TO P_1 FOLLOWS THE DOTTED PATH.
THE SENDING PROCESS P_3 WILL PASS THE SEND INFORMATION TO P_4 ,
THE NEW SENDING PROCESS.

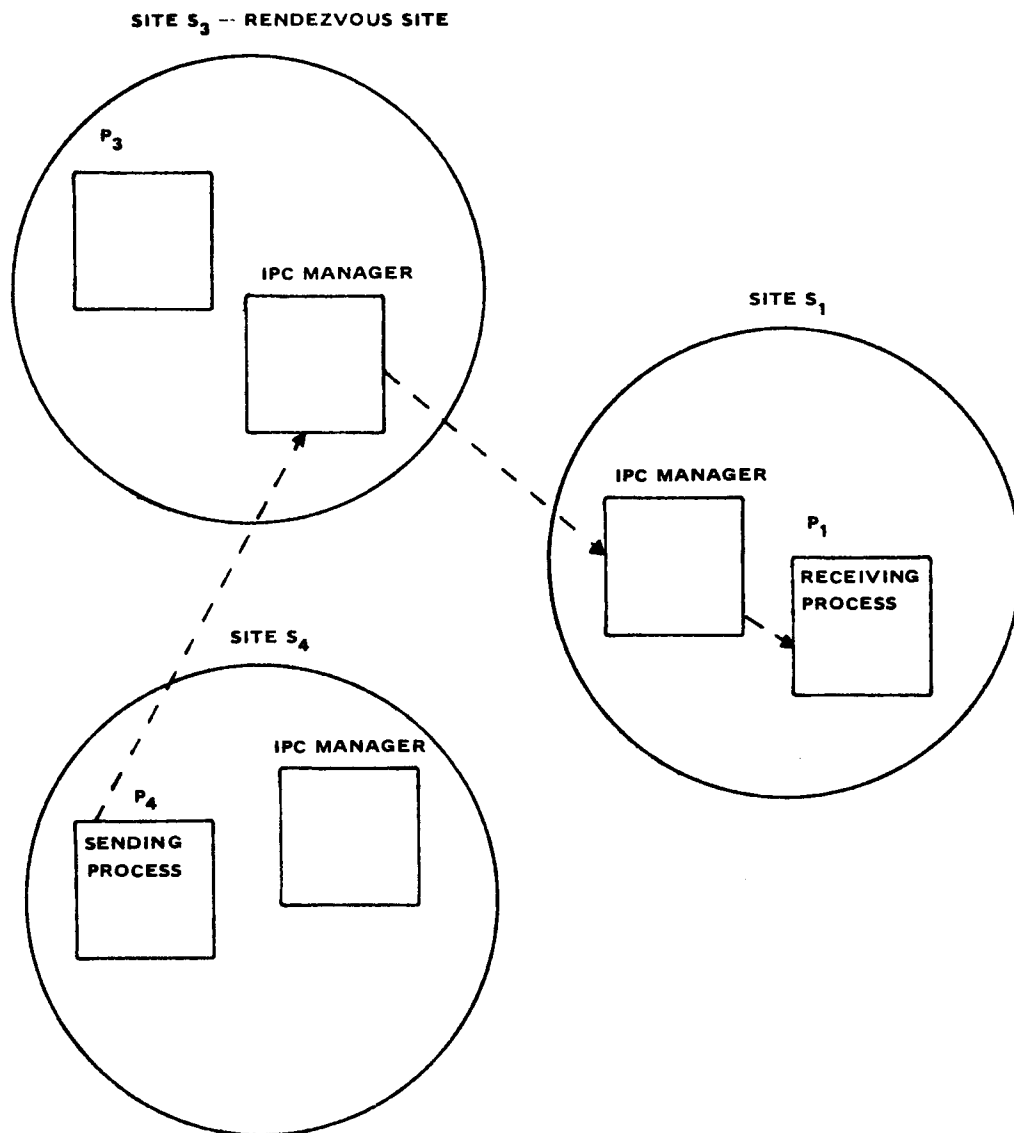


FIGURE 5. THE FIRST MESSAGE TRANSMISSION FROM P_4 TO P_1 FOLLOWS THE DOTTED PATH. THIS MESSAGE WILL SPECIFY S_4 AS THE RENDEZVOUS SITE FOR FUTURE MESSAGE TRANSMISSIONS FROM P_4 TO P_1 .

site S_3 and the message will be transmitted. The message will contain the rendezvous site number for future communications and any further RECEIVES can be executed using the new rendezvous site, S_4 (Figure 6).

Outstanding requests create no problem when channels are moved from one process to another process. Consider, first of all, an outstanding SEND request when a channel is moved from a receiving process to another receiving process. Suppose there exists a channel S from process P_1 to process P_2 , that is, P_1 has the information necessary to send a message to P_2 and P_2 has the information necessary to receive a message from P_1 . Suppose P_1 executes a SEND on channel S . Without executing a RECEIVE, P_2 can pass the channel to P_3 . P_3 can then execute a RECEIVE on channel S and the message from P_1 will be moved to P_3 .

Secondly, consider an outstanding SEND request when a channel is moved from a sending process to another sending process. Suppose there exists a channel S from process P_1 to process P_2 . Suppose that P_1 executes a SEND on channel S . Before P_2 executes a RECEIVE, P_1 can pass the channel to P_3 and P_3 can execute a SEND on channel S . Suppose P_2 then executes consecutive RECEIVES on channel S . If messages are sequenced, P_2 will receive the complementary messages from P_1 and the complementary message from P_3 as expected.

Similarly, outstanding RECEIVE requests create no

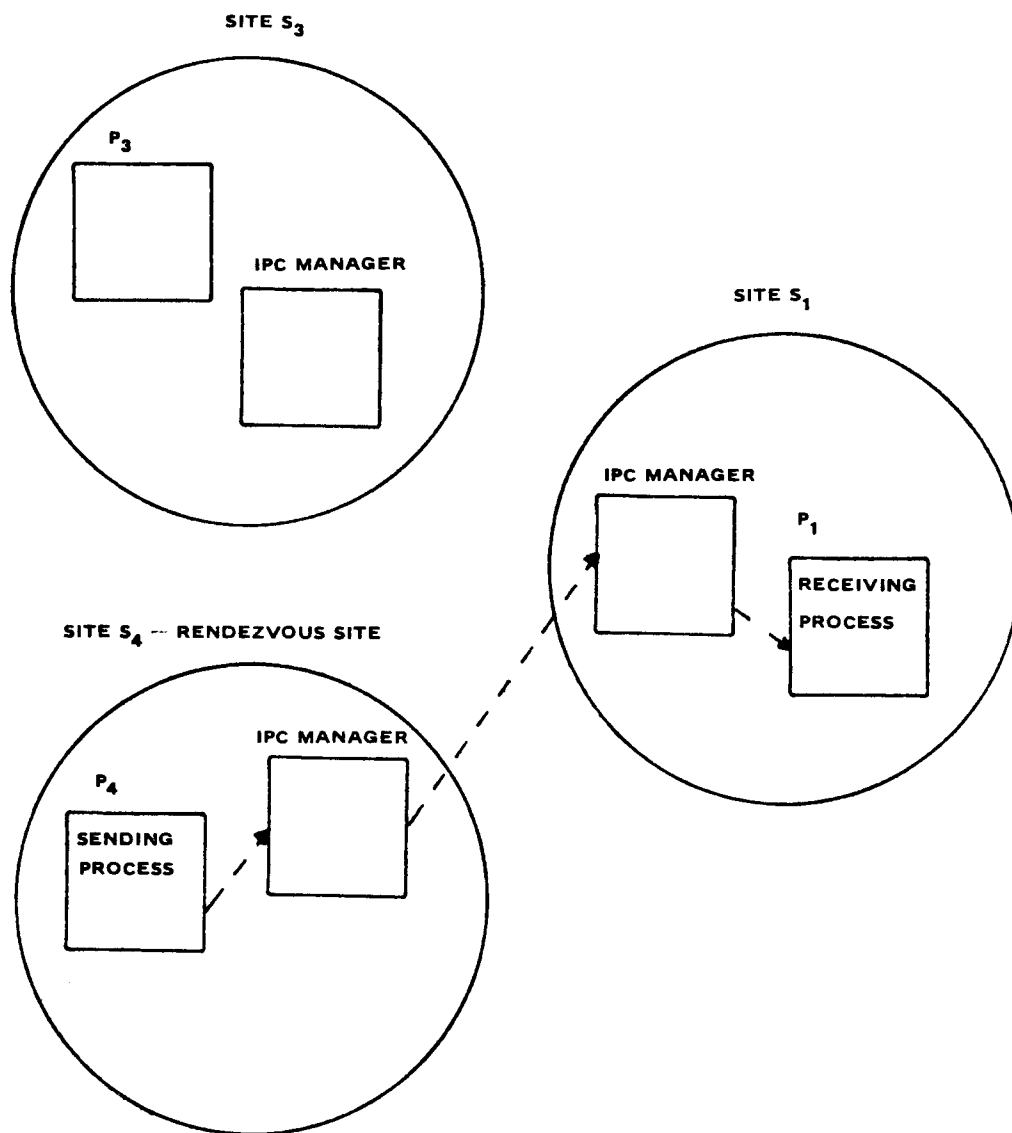


FIGURE 6. AFTER THE INITIAL MESSAGE TRANSMISSION FROM P_4 TO P_1 , S_4 WILL BECOME THE RENDZVOUS SITE FOR MESSAGE TRANSMISSIONS FROM P_4 TO P_1 . MESSAGE TRANSMISSIONS WILL FOLLOW THE DOTTED PATH.

problem when channels are moved from one process to another.

Unmatched SENDs and RECEIVES create no serious problem. If a SEND is executed and the complementary RECEIVE is not executed within a certain amount of time, the sending process is notified by an error response. Similarly, if a RECEIVE is executed and the complementary SEND is not executed within a certain amount of time, the receiving process is notified. In either case the notified process can then re-execute the operation if necessary.

MULTIPLE SENDERS AND RECEIVERS

There are no limitations in this facility on the number of senders or the number of receivers possible on a single channel. Multiple senders allow a process to receive a message from one of several senders. Multiple receivers allow a process to send a message to one of several receivers. It is not possible to specify a particular sender or a particular receiver. Note that it is possible to have multiple senders and multiple receivers on a single channel.

Since this facility does not preclude the possibility of multiple senders and multiple receivers, there are a few points to be considered. First of all, ordering of messages has no particular meaning in this context. Secondly, there is not a general solution to the establishment of the rendezvous site in remote communications. It was mentioned

earlier that when remote processes communicate, the rendezvous site is normally the site of the sending process. This convention was introduced so that it would be possible to block message transmission until the receiving process is ready to receive the message. In the case of one sender and multiple receivers, this convention is still appropriate. In the case of multiple senders and one receiver, having the rendezvous site at the site of the receiver may reduce the overhead that can result from matching SENDs and RECEIVES at a site of one of the senders. In the case of multiple senders and multiple receivers in remote communications, the user programmer must establish appropriate conventions.

AN ILLUSTRATION OF THE USE OF THE MESSAGE PASSING FACILITY

The following is a scenario of an operating system which will demonstrate the use of the Message Passing communication facility.

During initialization of the operating system an Input/Output device process is created for each device (terminal) with which a user can enter the system. Each device process is initialized with an output channel to SEND information to other processes and an input channel to RECEIVE information from other processes.

Also, during initialization of the operating system, a logger process is created and initialized with channels to communicate with the system. Through this communication channel the system passes channel numbers and the

information necessary for the logger to execute a RECEIVE on the output channel of each device process.

The device process SENDs the data typed in at the terminal on its output channel. Initially, the data will be sent to the logger process for validation of the identity of the user. The logger process then creates a command interpreter process. The device process channel numbers are passed by the logger to the command interpreter through creation-time parameters.

At this point the logging protocol is complete and the command interpreter is ready to receive commands. The command interpreter executes a RECEIVE on the device process output channel and as a user types in at the terminal the data is sent to the command interpreter. The command interpreter interprets commands as requests to use or create dataflow managers, or processes, and to redirect the paths of communication between processes by moving channels.

LOGGER PROCESS USING THE MESSAGE PASSING FACILITY

The logger process (Figure 7) is a dataflow manager that validates the identity of users who attempt to log-on to the system. The logger is initialized with a channel to receive messages from the system and a channel to receive log-off messages from command interpreters.

The logger executes RECEIVES to the system which result in a stream of messages, X. The system sets up channels between the logger and each device process and sends the

```

logger_process ← manager(cn_sys, st_sys, sn_sys, valid_msg, retry_msg,
                           cn_log, st_log)

(X ← (initial cn_sys, st_sys, sn_sys ← cn_sys, st_sys, sn_sys
      while true do
        x ← RECEIVE (cn_sys, st_sys, sn_sys);
        new sn_sys ← 1+sn_sys
        return all x);

Y ← merge (X, RETRY, FREE);

RETRY ← (for each y in Y do
  login_msg ← RECEIVE (y[cn_out], y[st_out], y[sn_out]);
  new y[sn_out] ← 1+y[sn_out];
  response, new y[sn_in], interpreter, retry ←
    (if validate (login_msg)
     then SEND (y[cn_in], y[st_in], y[sn_in], valid_msg)
           1+y[sn_in],
           create (command_interpreter, y[cn_out], y[st_out],
                   new y[sn_out], y[cn_in], y[st_in],
                   new y[sn_in], cn_log, st_log),
           λ
     else SEND (y[cn_in], y[st_in], y[sn_in], retry_msg),
           1+y[sn_in],
           λ,
           new y)
  return all retry but λ);

Z ← (while true do
  z ← RECEIVE (cn_log, st_log)
  return all z);

FREE ← (for each z in Z do
  response ← SEND (z[cn_in], z[st_in], z[sn_in], accounting(z));
  new z[sn_in] ← 1+z[sn_in];
  free ← new z when response
  return all free)      !end of logger process!

```

Figure 7

channel numbers to the logger.

For each device process in the system, indicated by the stream X, the logger executes a RECEIVE. The device process SENDS a message to the logger, log-in-msg, containing the information the user has typed in at the terminal. If the user's identity is valid the logger creates a command interpreter, passing the channel numbers of the device to the command interpreter. The command interpreter is also initialized with a channel, cn-log, to send log-off messages to the logger when the user is ready to log-off the system.

If the user's identity is invalid, a stream of the messages containing channel information, RETRY, is merged with the system stream, X. And the logger will execute another RECEIVE to the device process for another attempt.

When a user is ready to log-off the system, the command interpreter sends a message to the logger who has a RECEIVE request pending. The message sent by the command interpreter contains the channel numbers of the device process. The logger then uses these channels to send accounting and log-off information to the user. The stream of messages containing channel information of users who have logged off the system, FREE, is merged with the system stream, X, and the retry system, RETRY. And the logger will execute another RECEIVE to the device which is now available for a user to log-on to the system.

INTERPROCESS COMMUNICATION MANAGER FOR THE MESSAGE PASSING FACILITY

The interprocess communication manager (Figure 8) performs the actual communication operations. The manager is initialized with a site number, an initial unique number, and the state of the local communication channels.

Each operation request enters the IPC managers REQUESTQ. Each request is a structure value that contains the type of operation and the parameters necessary to perform the operation.

For unique number requests, sequential numbers are computed, and a unique number is returned as the result of the request.

For SEND and RECEIVE requests, the site number is checked to determine if the request is to be matched locally or to be sent to another site via the send expression. The state of each channel is represented by the structure value CHANNELS. And since each of these requests changes the state of a channel, a new structure is created which represents the new state of the channels in the manager. For the SEND operation the site number is returned as a result of the request. For the RECEIVE operation a message is returned.

```

ipc ← manager (site_no, unique, CHANNELS)
  (entry request: REQUESTQ do

    RESULT ←

      (initial site_no, unique, CHANNELS ← site_no, unique, CHANNELS
        for each x in REQUESTQ do

          result, CHANNELS ←
            (case

              x[request] = "send" ⇒

                (if x[st]=site_no
                  then (if x[sn]=nil
                    then (site_no, CHANNELS+[x[cn], x[m]])
                      when isnil CHANNELS[x[cn]]=true
                    else site_no, CHANNELS+[x[cn], x[sn], x[m]])
                  else send (x[sn], x), CHANNELS+[x[cn], x[sn], nil])

              x[request] = "receive" ⇒

                (if x[st]=site_no
                  then (if x[sn]=nil
                    then (s ← CHANNELS[x[cn]]
                      when isnil CHANNELS[x[cn]]=false
                      return s, CHANNELS+[x[cn], nil] when s)
                    else (s ← CHANNELS[x[cn], x[sn]]
                      when isnil CHANNELS[x[cn], x[sn]]=false
                      return s, CHANNELS+[x[cn], x[sn], nil]
                      when s)
                  else send (x[sn], x), CHANNELS+[x[cn], x[sn], nil])

              x[request] = "unique" ⇒

                (new unique ← 1+unique return unique, CHANNELS)

            return all result)

    exit request: RESULT)

```

Figure 8

Chapter 3

THE LOGICAL ASSIGNMENT COMMUNICATION FACILITY

BACKGROUND

Id provides a means of transmitting data between processes through the use expression. By writing,

$$z \leftarrow \underline{\text{use}} (m.p, y)$$

the input value y is sent to the entry port p of manager m and the result produced by manager m is returned as the value of the use expression [AGP78]. This mechanism provides a basis for a logical communication facility as follows. Suppose Process A wishes to communicate with other processes by executing a use to a logical entry port, "porta." Process A (or any other process) could assign a physical entry port, $m1.p1$, to "porta." Then any use expression to "porta" could be converted by a communication facility into a use of the physical entry port $m1.p1$. The communication facility could also provide a means of changing the logical entry port assignment. The assignment of "porta" could be changed from $m1.p1$ to $m2.p2$. A use expression to "porta" would then result in a use expression to the physical entry port $m2.p2$.

In the Logical Assignment Facility described here, there are provisions for ordering use operations. As the equivalent of multiple senders in the Message Passing Facility, there may be multiple copies of a logical entry

port name in the computer system. Similarly, as the equivalent of multiple receivers in the Message Passing Facility, there may be multiple physical entry ports assigned to a single logical entry port name. Also, there are provisions for remote as well as local communication.

DESCRIPTION

The following operations provide a means of logical communication in a dataflow system. The operations are carried out at each site within the computer system. A dataflow manager, referred to as an Interprocess Communication (IPC) manager, performs the following operations:

1. ASSIGN. This operation assigns a physical entry port to a logical entry port. The operation has the following parameters:
 - a. Logical Entry Port Name.
 - b. Physical Entry Port. An entry port of an instance of a manager, given as m.p where m is the manager instance and p is an entry port of m.
 - c. Sequence Number (optional). A value indicating an initial ordering number.
2. CHANGE. This operation changes an assignment to a logical entry port from the currently assigned physical entry port, m1.p1, to a newly assigned physical entry port, m2.p2. The operation has the following parameters:

- a. Logical Entry Port Name.
 - b. M1.p1. The currently assigned physical entry port.
 - c. M2.p2. The physical entry port to replace the assignment of m1.p1.
 - d. Sequence Number (optional). A value indicating an initial ordering number.
3. USE. This operation results in a use of a manager to a physical entry port currently assigned to the given logical entry port. Arguments to the manager are passed as a parameter in the USE operation. The parameters in the USE operation are as follows:
- a. Logical Entry Port Name.
 - b. Arguments. The arguments to be sent to the manager in the use expression.
 - c. Sequence Number (optional). A value indicating the sequence of the use expression.
4. UNIQUE. This operation obtains a unique name from the communication facility.

This facility can be used as follows. Suppose Process A wishes to communicate with other processes by executing a USE to a logical entry port, "porta." A unique logical entry port name such as "porta" can be obtained from the communication facility via the UNIQUE operation. Process A

(or any other process) initially executes an ASSIGN on "porta" with physical entry port m1.p1. Any USE operation executed on "porta" with argument Y will result in a use of manager m1 to entry port p1 with an input value Y (Figure 9). At any point after the initial assignment of "porta," a CHANGE can be executed on "porta" with physical entry port m1.p1, and physical entry port m2.p2 as parameters. Until another ASSIGN or a CHANGE is executed on "porta," any further USE operation on "porta" will result in a use of manager m2 to entry port p2 (Figure 10).

In the CHANGE operation, "null" can be given as the parameter defining the new physical entry port. The effect of such an operation is to delete a physical entry port assigned to a logical entry port name without making a new physical entry port assignment.

Each of the operations, ASSIGN, CHANGE, USE, and UNIQUE is implemented as a procedure call with the parameters of the operation passed as arguments to the procedure. The procedure creates a structured parameter and executes a use of the local IPC manager. In the ASSIGN and CHANGE operations, the result returned from the IPC manager is an acknowledgment which may be an error response. In the USE operation, the result returned from the IPC manager is the result of the use of a manager assigned to the specified logical entry port name. In the UNIQUE operation, the result returned from the IPC manager is a unique name.

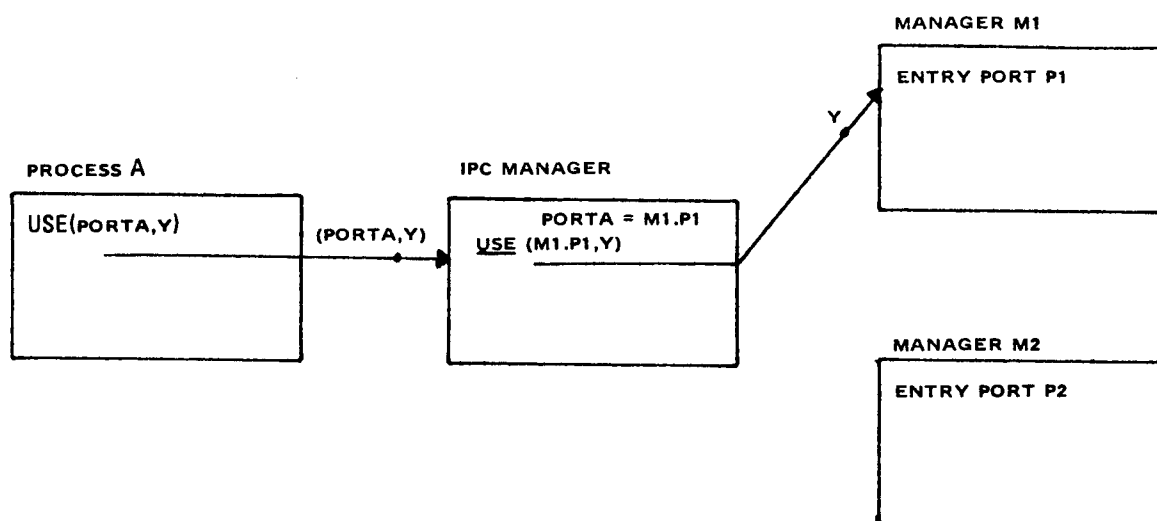


FIGURE 9. A USE EXECUTED ON "PORTA" WILL RESULT IN A USE TO ENTRY PORT P1 OF MANAGER M1.

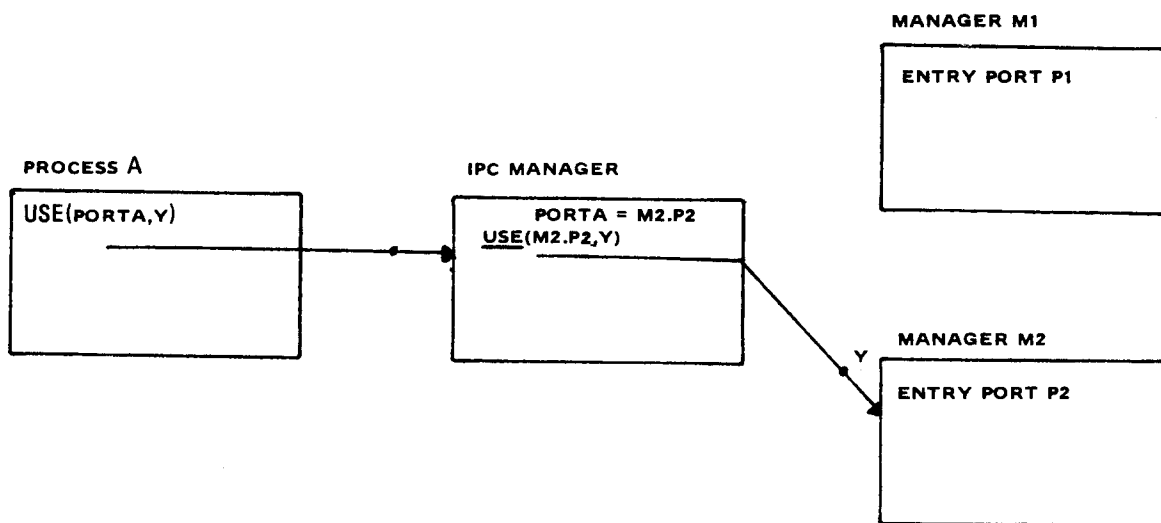
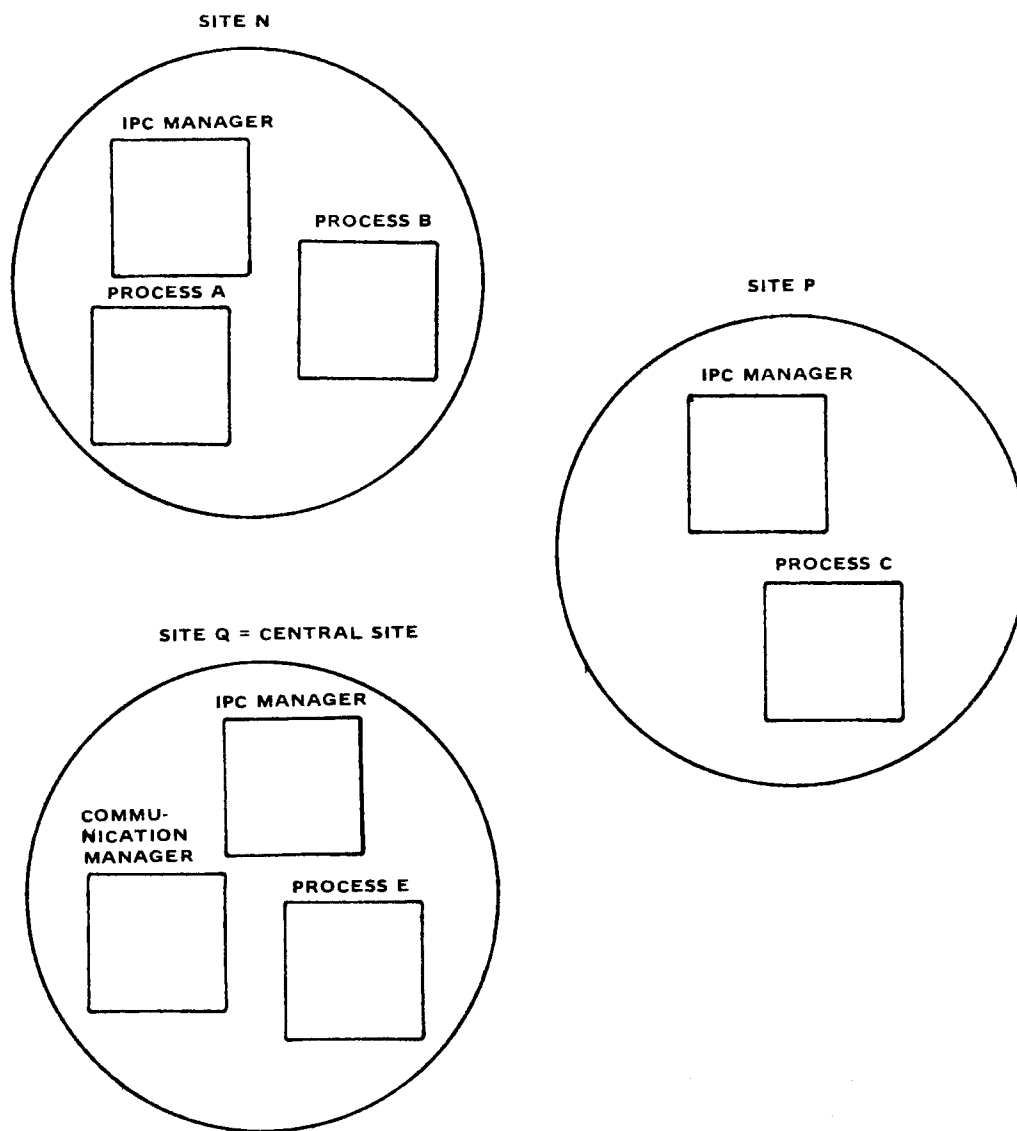


FIGURE 10. AFTER M1.P1 IS CHANGED TO M2.P2, A USE EXECUTED ON "PORTA" WILL RESULT IN A USE TO ENTRY PORT P2 OF MANAGER M2.

In order to provide communication between remote processes, there exists an IPC manager at each site in the computer system. At one of the sites, called the central site, there exists a dataflow manager, referred to as the Communication manager (Figure 11). Within the computer system there must exist the ability to transmit a request to a remote site. How this is done is not important to this discussion. Therefore, as in the Message Passing Facility, there exists a send expression that transmits a request to a remote site and returns the result of the request. The expression has two parameters: site number (the identifying number associated with the site where the request is to be transmitted) and the request.

The Communication manager maintains at the central site all information associating logical entry port names with physical entry ports and provides copies of logical entry port name assignments to the IPC managers. When the Communication manager receives a request from an IPC manager for a logical entry port name assignment, the Communication manager will return a copy of the logical entry port name assignment and record the site number of the location of the copy. When the Communication manager receives a request from an IPC manager to assign or change a logical entry port name, the Communication manager will perform the operation at the central site and then send the request to each IPC manager where a copy of the assigned or changed logical



**FIGURE 11. AN IPC MANAGER IS LOCATED AT EACH SITE WITHIN THE COMPUTER SYSTEM.
THE COMMUNICATION MANAGER IS LOCATED AT THE CENTRAL SITE.**

entry port name is located. When all of the copies have been updated the result of the assign or change request will be returned to the IPC manager that originated the assign request.

Each UNIQUE, USE, ASSIGN, and CHANGE operation is initially sent to the local IPC manager of the process executing the operation. With the UNIQUE operation, the local IPC manager will return a unique name. With the USE operation, the local IPC manager will check if it has a copy of the logical entry port name assignment. If it has a copy the local IPC manager will execute the use and return the result. If the IPC manager does not have a copy of the logical entry port name assignment, the IPC manager will request a copy from the Communication manager at the central site via the send expression. After receiving a copy of the logical entry port name assignment, the local IPC manager will execute the use and return the result. With the ASSIGN or CHANGE operation, the local IPC manager will send a request to the Communication manager to assign or change the logical entry port name. When the result of the send expression is received by the local IPC manager, the assign or change request will have been performed at the central site and at each site where a copy of the logical entry port name is located. The result of the request at the central site will be returned as the result of the ASSIGN or CHANGE operation.

The location of a physical entry port will be encoded in the physical entry port name. When an IPC manager performs a USE operation, the IPC manager will determine if the physical entry port is located at a remote site. If the physical entry port is not at a remote site but at the same site as the IPC manager, the IPC manager will perform the use. If the physical entry port is located at a remote site, the IPC manager will send the use request to the IPC manager at the remote site to be performed.

The user programmer is responsible for coordinating CHANGE, ASSIGN, and USE requests to avoid problems with outstanding requests, i.e., those requests that may be "in the pipe" when a new assignment is made to a logical entry port name. USE or CHANGE requests to a non-existent logical entry port create no serious problem. The requesting process will be notified of the error by an error response. The notified process can then take whatever corrective action is necessary. And after an assignment is made to the previously non-existent logical entry port, the notified process can re-execute the USE or CHANGE operation.

ORDERING OF USE OPERATIONS

In a dataflow system requests to execute a use to a physical entry port may arrive at the site of the IPC manager performing the operation in an indeterminate order. For example, this might result from an unraveled loop containing a USE operation. Consequently, it may be

necessary to order the execution of several sequential USE operations. Ordering is provided by the sequence number (an optional parameter).

If ordering is required, the user programmer is responsible for providing an initial sequence number as a parameter in the initial assignment of a physical entry port to a logical entry port name. The user programmer is then responsible for incrementing the sequence value and providing it as a parameter in USE operations on the logical entry port.

If the sequence number S is provided as an initial sequence number in an ASSIGN or CHANGE operation, the use resulting from a USE operation with a sequence number of S will be initially executed. After the result of the initial use is returned to the site of the IPC manager performing the operation, the use resulting from a USE operation with a sequence number of $S+1$ will be executed.

If a sequence number is not provided in an ASSIGN or CHANGE operation, no ordering is provided. Similarly, if a sequence number is not provided in a USE operation, no ordering is provided by the communication facility.

This facility places no restrictions on the number of processes that can execute a USE to a single logical entry port. A process can pass a logical entry port name to another process and there can be several copies of a logical entry port name in the system at one time.

Similarly, the facility places no restriction on the number of physical entry ports that can be assigned to a single logical entry port. Executing a USE operation to a logical entry port with more than one physical entry port assignment allows a use to be executed to one of the assigned physical entry ports. A scheduling algorithm could be supplied to the communication facility for a logical entry port name. This algorithm would determine which of several assigned physical entry ports would be used in each USE operation to the logical entry port.

AN ILLUSTRATION OF THE USE OF THE LOGICAL ASSIGNMENT FACILITY

The following scenario of an operating system demonstrates the use of the Logical Assignment communication facility.

During initialization of the operating system an Input/Output device process is created for each device (terminal) with which a user can enter the system. Each device process is initialized with a unique logical entry port L. The logical entry port L is used by the device process to send data that is input to the terminal and to receive data to be output to the terminal. As a user types in at a terminal, the output data is sent as an input value to the logical entry port L via the USE operation. The logical entry port L is initially assigned to a physical entry port of the

logger process. After validation of the user's identity, the logger creates a command interpreter process. The logger process then executes a CHANGE operation to change the physical entry port assigned to L from the entry port of the logger to an entry port of the command interpreter. At this point any further data input the device process is sent to the appropriate physical entry port of the command interpreter. The command interpreter interprets commands to create and/or use dataflow managers. In doing so, the command interpreter sets up the appropriate logical communication assignments.

LOGGER PROCESS USING THE LOGICAL ASSIGNMENT FACILITY

The logger process (Figure 12) is a dataflow manager which validates the identity of users who attempt to log-on to the system. The logger is also responsible for returning accounting information when a user is ready to log-off the system. The logger has two physical entry ports that correspond to these functions.

If a user's identity is valid the logger process creates a command interpreter dataflow manager for the user. The device process associated with the user sends input data to a logical entry port, L. The validated entry port of the logger process was initially assigned to L. After validation of the user's identity, the logger process changes the assignment of L from the validate entry port of the logger to the input entry port of the command interpreter. This

```

logger_process ← manager (valid_msg, retry_msg)

(entry validate: VALIDATEQ;
 terminate: TERMINATEQ do

  LOG_IN_RESULT ←
    (initial valid_msg, retry_msg ← valid_msg, retry_msg
     for each v in VALIDATEQ do
       response, interpreter, result ←
         (if validate (v)
           then create (command_interpreter, v[l_name]),
             CHANGE (v[l_name], logger_process.validate,
               interpreter.input_port),
             valid_msg
           else λ,
             λ,
             retry_msg)
       return all result);

  LOG_OFF_RESULT ←
    (for each t in TERMINATEQ do
      return all accounting (t));

exit validate: LOG_IN_RESULT
      terminate: LOG_OFF_RESULT)

```

Figure 12

causes any further inputs from the terminal to be sent to the created command interpreter.

If a user's identity is invalid the logger process returns a retry message to the user. The device process associated with the user will continue to send input data to the logger for future attempts by the user.

When the user is ready to leave the system, the command interpreter will change the assignment of logical entry port L from the input entry port of the command interpreter to the terminate entry port of the logger. The logger process will return any accounting information to the user.

Chapter 4

A COMPARISON AND EVALUATION OF THE COMMUNICATION FACILITIES

Neither of the communication facilities discussed in this thesis has been implemented. Although operational performance conclusions cannot be made, some observational comparisons are possible.

The Message Passing and Logical Assignment communication facilities discussed in this thesis are similar in several respects. Both facilities provide a means of logical communication, allow multiple senders and multiple receivers, provide a mechanism for remote, as well as local, communication, and allow the communication to be set up by the processes themselves. There exists no inherent restriction in either facility on the amount or type of data that can be transmitted between processes.

In the Message Passing Facility the receiver has the opportunity to stop the flow of data after each data transfer. A message remains at the rendezvous site until the receiving process executes a RECEIVE. This can be accomplished in the Logical Assignment Facility by allowing the receiver to establish the scheduling algorithm for a logical assignment. The scheduling algorithm can be utilized to stop the data at the site of the IPC manager. Also, a user might establish a simple "kernel" manager that then does its own scheduling; it is thus an "active buffer process." The

input to a particular physical entry port can either be restricted at the site of the IPC manager or diverted to another physical entry port assigned to the same logical entry port.

In the Logical Assignment Facility it is possible to execute one operation, a USE, to send data to another process and to receive data from that same process. It is possible to create DSEND and DRECEIVE operations to provide a similar capability in the Message Passing Facility. A DSEND could be implemented to allow a process to send data to another process and then to receive data from that same process as the result of the operation.

Moving a sender from one process to another process is comparable in the two facilities. In the Message Passing Facility the move involves passing the information necessary to execute a SEND. In the Logical Assignment Facility, moving a sender from one process to another process involves passing the information necessary to execute a USE. In the Logical Assignment Facility a CHANGE is executed to move a receiver. This move requires only one process, the process executing the CHANGE. In the Message Passing Facility, moving a receiver can be done through creation-time parameters. However, passing the move information in a message requires the coordination of both the old receiver and the new receiver.

In both the Message Passing Facility and the Logical Assignment Facility it is possible to order the data that

will be sent to a receiver. Ordering is more easily accomplished in the Logical Assignment Facility, because only the sender is required to participate in the ordering. In the Message Passing Facility coordination between the sender and the receiver is required to accomplish an ordering of the messages.

These facilities differ in several respects. In the Message Passing Facility messages are set up and expire on a message by message basis. For each transfer of data one control message, the RECEIVE, must move from the receiving process to the rendezvous site for the data to move from the sender to the receiver. In the Logical Assignment Facility connections are set up once and they do not expire until a process terminates the connection. One control transmission is required to set up a connection - the ASSIGN or CHANGE operation. The connection exists until another CHANGE or ASSIGN is executed on the particular logical entry port name. After the initial connection is made, data can move from the sender to the receiver with each execution of the USE operation.

Another difference in the facilities relates to multiple receivers. In the Message Passing Facility a process cannot specify which of several receivers it will communicate with. However, it is possible to make this specification in the Logical Assignment Facility. This can be done by provision of a scheduling algorithm that will

determine which of several receivers will receive the data.

In summary, the Message Passing Facility and the Logical Assignment Facility are similar in most respects. However, for several reasons the Logical Assignment Facility appears to be more favorable. In the Logical Assignment Facility there is less coordination required between processes in passing move information and in ordering data. There is less overhead in the Logical Assignment Facility in communications that exist for more than one data transfer. The Logical Assignment Facility is more flexible than the Message Passing Facility in the area of multiple receivers since it is possible to specify which of several receivers will receive the data. Also, the Logical Assignment Facility follows more closely with the conventions that have been established for the Id language making it easier to utilize.

References

- [AGP78] Arvind, Gostelow, K. P., and Plouffe, W. E. An Asynchronous Programming Language and Computing Machine. Tech. Report 114A, Department of Information and Computer Science, University of California, Irvine, December, 1978.
- [Backus78] Backus, J. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. Comm. ACM 21, 8 (Aug. 1978), 613-641.
- [Brinch Hansen70] Brinch Hansen, P. The Nucleus of a Multiprogramming System. Comm. ACM 13, 4 (April 1970) 238-241.
- [Dennis, Van Horn66] Dennis, J. B. and Van Horn, E. C. Programming Semantics for Multiprogrammed Computations. Comm. ACM 3,9 (1966), 143-155.
- [Dennis73] Dennis, J. B. First Version of a dataflow procedure language. Computation Structures Group Memo 93, Lab. for Computer Science, Cambridge, Massachusetts, Nov, 1973.
- [GIMT74] Glushkov, V. M., Ignatyev, M. B., Myasnikov, V. A., and Torgashev, V. A. Recursive machines and computing technology. In Information Processing 74, J. L. Rosenfeld, Ed., North-Holland, New York, 1974.
- [Hoare78] Hoare, C.A.R. Communicating Sequential Processes. Comm. ACM 21, 8 (August 1978), 666-677.
- [Hunt78] Hunt, J. G. Messages in Typed Languages, ACM SIGPLAN Notices, 14, 1 (January 1979), 27-45.
- [RT74] Ritchie, Dennis M. and Thompson, Ken. The Unix Time-Sharing System. Comm. ACM 17, 7 (1974), 365-375.
- [Walden72] Walden, David C. A System for Interprocess Communication in a Resource Sharing Computer Network. Comm. ACM 15(4), (April 1972), 221-230.

Appendix A

THE DESTRUCTION OF DATAFLOW MANAGERS

The destruction of a dataflow manager [AGP78] should take place when the manager is no longer in use. Determining that a manager is no longer in use can be done with a reference-counter scheme. This reference-counter scheme would involve keeping a count of the number of references to each dataflow manager and destroying managers whose reference-counts drop to zero. However, circular name references are possible among dataflow managers, so the reference-counter scheme will not always detect all of the managers eligible for destruction. Therefore, in addition to a reference-counter scheme, there must exist a scheme for detecting circular name references among dataflow managers.

The Logical Assignment Facility provides the potential for solving the problem of detecting circular name references among dataflow managers. With modification to the Logical Assignment Facility, all of the information required to detect circular name references could be made available to the IPC manager. This can be done by removing the create and use expressions from the user and replacing them with CREATE and USE operations performed by the IPC manager. Name references to dataflow managers are logical entry port names. The IPC manager could know all name references of each dataflow manager, because the only

methods of acquiring logical entry port names are as the result of the UNIQUE operation, as a creation-time parameter in the CREATE operation, or as an input value in the USE operation.

In addition to the reference-counter scheme, a circular reference detection algorithm could be implemented by the IPC manager every time T to determine if there are any other dataflow managers eligible for destruction. The implementation of a detection algorithm and the modifications to the Logical Assignment Facility are areas that remain to be investigated.