[REDACTED]
CERN-Data Handling Division
DD/78/5
May 1978

CM-P00059734

A MESSAGE DISTRIBUTION TECHNIQUE
AND ITS APPLICATION TO NETWORK CONTROL

Roar A. Fjellheim

SUMMARY

The patterns of message exchange in distributed computer systems can become sufficiently complex to justify the construction of communication services that extend the basic message transmission mechanism. A simple method for implementing a copy distribution, or broadcast, service is described. It is shown how the method can support command and monitoring functions in a computer communication network.

KEYWORDS

Distributed systems, Message techniques, Network control

Submitted to "Software - Practice and Experience"

INTRODUCTION

Process-structured computer systems are often shown as in Figure 1, where a number of concurrently executing sequential processes interact through an interprocess communication mechanism (IPCM). Much effort has gone into finding IPCMs well suited to system design and implementation¹. Proposed mechanisms vary considerably, both in general approach and in detail. This is hardly surprising, since systems themselves have widely different characteristics, in terms of geographical dispersion of processes, interaction volume and frequency, etc.

This paper deals with systems using process-to-process message transmission as the basic communication mechanism. The general principle is quite simple. A source process creates a message, including the name of the destination process, and sends it across its interface to the IPCM. The IPCM transports the message to the destination, where the destination process receives it across its interface. It should be noted that many important questions, such as message buffering, interface design, etc. have been glossed over in this simplified account.

A message-transmitting IPCM is a basic tool for process interaction, but in complex systems more sophisticated facilities may be required. The model of Figure 2, showing a communication service inserted between each process and the IPCM, is appropriate for at least some IPCM enhancements. In message systems, a communication service intercepts messages to and from its associated process, and can treat them in any way desired. A service could generate traffic statistics, file away copies of all messages for recovery purposes, filter or reorder messages, or simply pass them on. Many systems of this kind exist, including terminal assistance facilities, computer mail systems², and sophisticated 'user agents'³.

The sections that follow present a simple message distribution service, particularly suited to systems where one to many, or broadcast, message transmissions are normal. The service is first defined by a general implementation, and by a notation for using it. A particular implementation in a computer network is then described, and finally some examples are offered on how the implemented service is used.

A MESSAGE DISTRIBUTION SERVICE

The particular communication service to be described is called a message distributor (MD), and is designed for the management of single-source, multiple-destination message transmissions. A message distributor intercepts all messages crossing the interface between a process and the IPCM. An informal description of its behaviour is first given.

A message entering the MD from its associated process contains a list of destination names. A copy of the message body is sent to each destination, except if the name is a special one that denotes a distribution list of process names maintained by the MD on behalf of the process. In the latter case, a copy is sent to each process currently on the list. A message addressed to the process, entering the MD from the IPCM or the process itself, may contain a command to update the distribution list. Such a command is obeyed, but the message is not forwarded. It should be noted that the messages injected into the IPCM contain a single process destination; thus no broadcast facility in the IPCM is assumed.

In order to give a more precise specification, some notation is needed. A message has the two components dest, a list of one or more destination names; and info, the information content of the message. To select the two components from a message m, the notation m.dest and m.info is used. A message is composed by enclosing the two components within angle brackets. The dest field may contain the special values list, indicating the distribution list, or self, standing for the process (and its MD) handling the message. The info field contains data meant for either the process or for its associated MD. It must be possible to distinguish between the two cases, through a special flag contained in the field, a particular pattern, or some other method.

A message distributor, associated with a process (self), can now be defined by the following three attributes.

- a. A distribution list dlist of destination process names. Two operations with obvious interpretations are defined on the list :

```
insert(process names)
remove(process names)
```

The insert operation ensures that a particular process name is not duplicated in the list, and that self is not entered into it.

- b. A message processing procedure. Using an informal programming language, the procedure can be described as follows :

```
procedure message(m) is
  if m.dest contains a single name
  then if m.dest is self
    then if m.info is a dlist update
      then update the list
      else pass m to the process
    else if m.dest is list
      then call message(<dlist,m.info>)
      else pass m to the IPCM
  else for each name in m.dest
    do call message(<name,m.info>)
```

- c. A procedure invocation rule stating that :

For each message m entering the MD from the process or from the IPCM, the statement 'call message(m)' is executed. The calls are strictly serial, to avoid inconsistent states of the distribution list. One call, possibly including recursive activations, must complete before the next is begun. This might require some buffering capability in the MD.

A NOTATION FOR MESSAGE DISTRIBUTION

A notation for describing sets of process names is needed in order to have concise commands for message transmissions. Only two simple forms will be used here, but others are conceivable.

Explicit enumeration of names : proc1,proc2,..

The universe of all names : all

For systems with composite process names, the notation can be applied independently to the name components. For example, if the names have the form a.b, one could have sets a.(b1,b2), all.b, all.all, etc.

The interface between a process and the IPCM must at least contain primitives for sending and receiving messages. The following syntax will be assumed, where 'receivers' is a process name set.

send data to receivers
is equivalent to transmitting <receivers,data>

send data
is equivalent to send data to list

receive data
is equivalent to receiving <self,data>

This is clearly not a complete specification of a process-IPCM interface; however, it will suffice for the present exposition.

A command to update the distribution list may originate from any process, and the decision to start or stop a copy-stream of a process' message transmissions can be taken by that process itself, by the recipient of the copies, or by a third party process. The following notation will be used to describe such copy decisions, where 'senders' and 'receivers' are process name sets.

copy on from senders to receivers
means send insert(receivers) to senders

copy off from senders to receivers
means send remove(receivers) to senders

Two short-hand forms of the copy on command are accepted.

copy on from senders
means copy on from senders to self

copy on to receivers
means copy on from self to receivers

The copy off command has similar abbreviations.

After a copy on operation has completed, any command of the form 'send data', issued by a process in the senders set, will cause the transmission of at least the messages <proc,data> to all processes proc in the receivers set.

One way of utilizing this interface is to let processes normally communicate through 'send data' and 'receive data' commands, and leave the addressing of messages to the distribution services. Processes may then easily be designed to be independent of particular system configurations. Paths between message producers and consumers are set up dynamically, under the control of the partners themselves or supervisor processes.

IMPLEMENTATION IN A NETWORK

The message distribution technique described in the previous sections is used in the operation of CERNET, a packet switching network built on the CERN site⁴. CERNET connects a variety of computers, ranging from large mainframes to small data acquisition computers, via a network consisting of minicomputer switching nodes and high speed data links.

The operation of the network is based on fairly standard methods⁵. A routing scheme with a distributed updating algorithm ensures continued service through reconfigurations caused by node or link failures. A control centre monitors the state of the network and permits operator or program intervention whenever necessary. For redundancy, every node is capable of performing basic network control functions via operator actions on its own console.

Each node has a number of processes, for link protocol handling, link supervision, packet routing, etc. The processes expect commands from, and return responses to, an operator console. For uniformity, the console of a node is treated as a process, with the property that a line of typed console input causes the process to transmit a message, and a received message appears as printed output. If MD denotes message distributors, i.OP the operator console process, and i.PR1, i.PR2, .. other processes of node i, the node subnetwork can be shown as in Figure 3. The node and the IPCM overlap since any process (i.OP or other) of node i can address a message to any process of node j, including the case i=j. Thus console to console, console to program, and program to program communication on the same or on different nodes are all served by the same mechanism. In addition, the message distributors support one-to-many transmissions.

In the actual implementation, processes communicate through a relatively modestly sized extension to the basic console handling software of the node. The extension includes a message distributor facility, and supports inter-node communication through the normal packet carrying ability of the network. The messages are console-oriented character strings, and the process-IPCM interface is identical to the standard console read/write functions. Thus, in a sense the processes are treated as consoles, rather than the other way around.

NETWORK CONTROL SUPPORT

This section shows some examples of how the distribution service is used to support the operation of CERNET. For clarity, only the syntax developed in the section on notation for message distribution is used. However, compound commands could easily be defined, e.g. for combined path clearance and establishment, for setting up bi-directional paths, etc. Unless otherwise stated, all processes are assumed to use the 'send data' command for message transmission.

- a. The operator at the centre can broadcast a command to the processes SUB of all nodes, by entering

send command to all.SUB

Distributing the network time is one application of this feature. By replacing SUB with OP, consoles instead of programs will receive a copy of the centre command.

- b. The network has duplicated nodes in front of large host machines. When a host goes down, identical commands need to be given to the two front-ends, for example for dumping end-of-session statistics, for closing the links, etc. An operator can cause such a command to be sent to the process SUB of two front-ends i and j, by entering

send command to (i,j).SUB

- c. If a longer sequence of commands is to be given in the previous example, it is more appropriate first to enter

copy on to (i,j).SUB

and then continue to issue

send command

- d. An operator at a node i (say, the control centre) can 'attach' his console to process SUB of node j by entering

copy on to j.SUB
copy on from j.SUB

From now on, a copy of input typed on the centre console will be sent to process SUB of node j, and output from that process will be returned to the operator's console. By replacing SUB with OP, two consoles are attached together, for operator-to-operator dialogues.

- e. The process j.SUB in the example above may normally interact with a process SUP, instead of with the operator, at node i. The operator can switch it back to this normal mode by the commands

copy off to j.SUB
copy off from j.SUB
copy on from i.SUP to j.SUB
copy on from j.SUB to i.SUP

- f. The centre operator can monitor nodes i..j, or all nodes, by arranging to get copies of all console and program messages produced at these nodes, through entering one of

copy on from (i,...,j).all
copy on from all.all

Node traffic statistics, error reports, etc. are sent to the centre in this way.

- g. A process LOG at the centre that issued the same commands as in the previous example would be in a position to create a file of network events. Instead of letting the logging program itself connect to all other processes, the centre operator could set up paths, for example to LOG and to a printer process PRINT.

copy on from all.all to i.(LOG,PRINT)

- h. Should it be necessary to let node j become the centre instead of node i, the copy streams to LOG could be redirected by

copy off from all.all to i.LOG
copy on from all.all to j.LOG

CONCLUSION

The mechanism presented in this paper was designed for the particular purpose of network control, and it serves that purpose well for the CERN network. An attempt has been made to explain the relatively simple principle involved in more general terms, since it might be applicable to other environments, such as the 'automated office'.

ACKNOWLEDGEMENTS

The contributions of C.Bloch, J.M.Gerard, C.Piney, and J.Richter, all of the CERNET implementation team, to the ideas presented are gratefully acknowledged.

REFERENCES

1. Proceedings of the SIGCOMM/SIGOPS Interprocess Communications Workshop, Association for Computing Machinery, 1975
2. T.H.Myer and D.W.Dodds, 'Notes on the development of message technology', Proceedings of the Berkeley Workshop on Distributed Data Management and Computer Networks, 1976, p.144-154
3. R.H.Anderson and J.J.Gillooly, 'The RAND intelligent terminal agent (RITA) as a network access aid', Proceedings of the National Computer Conference Vol. 45, 1976, p.501-509
4. J.M.Gerard, 'CERNET, The CERN packet-switching network', CERN-Data Handling Division, DD/77/20, Nov. 1977
5. R.A.Fjellheim, 'Network control - Overview', CERN-Data Handling Division, DD/NPN/59, April 1978

Fig. 1 — PROCESS-STRUCTURED SYSTEMS

Fig. 2 — COMMUNICATION SERVICES

Fig. 3 — LOGICAL STRUCTURE OF NODE SUBNET