A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 · DK-2000 COPENHAGEN F

RCSL No: 31-D372
Edition: September 1975
Author: Ole Krag Hansen

Title: RCNET

General Description

Keywords: RC 4000, RC 8000, RC 3600, Computer Network, Device Controller, Communication Protocol.

Abstract: The manual contains general information about the Computer Network RCNET.

Contents

1.	Introduction	1.1
2.	Definitions and Concepts	2.1
3.	A network example	3.1
4.	Components of RCNET	4.1
	4.1 System Components	4.1
	4.2 RC 8000 Communications Software	4.2
	4.3 RC 3600 Device Controller Package	4.5
	4.4 RC 3600 Network Node Package	4.7
5.	Protocol levels in RCNET	5.1
6.	Line Control	6.1
	6.1 RC 4000 - RC 3600 / RC 3500	6.1
	6.2 RC 8000 - RC 3600 / RC 3500	6.2
	6.3 Network node - network node / Network node - RC 3600 Devicecontroller	6.2
7.	Packet Control	7.1
	7.1 Contents of link table entry	7.5
	7.2 NCC treatment of messages from a host	7.9
	7.3 Contents of Packet Control Information	7.9
	7.4 NCC treatment of incoming packets	7.10
	7.5 Contents of Packet Control Acknowledge	7.11
8.	Message Control	8.1
	8.1 Message Control Protocol for Device Control	8.2
	8.2 Contents of messages and answers	8.6
	8.3 Needs for other types of protocol	8.8
9.	Routing Principles	9.1
	9.1 Fixed Routing	9.2
	9.2 Shortest Path Routing	9.2
	9.3 Advantages of Dynamic Routing	9.5
	9.4 Delay Estimate	9.6
10.	Creation and Removal of links	10.1
	10.1 Contents of "create" Message	10.2
	10.2 "lookup" and "include" Messages	10.3

References

1. Ole Krag Hansen, Erik Lilholt
   RC 4000 - RC 3600 Synchronous Communication Protocol
   (In Danish) RCSL: 31-D355

2. Data communication - High-level data link control procedures -
   Frame structure. ISO/DIS 3309

3. IBM Synchronous Data Link Control
   General Information GA 27 - 3093 - 0

4. RC 4000 Monitor, Definition of External Processes
   RCSL: 31-D38-D62

1. Introduction

RCNET is a computer network, which is intended to be used as basis for implementation of communication systems.
The purpose of this manual is to give a broad description of the facilities and principles of RCNET.

Some of the central design criteria are:

1. Host computers of different types may be connected without changing host computer software. Efficiency in use of the network may however be increased by use of special software in a host.
2. Standard conventions should exist for implementation of new services at different network levels.
3. Smooth degradation in overall performance in case of component failure. This is achieved by distributed intelligence, so the network does not rely on a central network supervisor.

The basic design includes the following functional components:

1. A transport subnet of network nodes connected by communication lines. The transport subnet accomodates changing network conditions by means of a distributed, dynamic routing algorithm.
2. A general purpose devicecontroller based on the RC 3600 support system. It takes over the functions performed by units known as multiplexors, concentrators, cluster controllers, block multiplexor etc.
3. Communications software in the RC 8000 computer, enabling user programs to communicate with remote devices and remote user programs by the access methods used in communication with local devices and local user programs. Also, remote users have readily access to the great processing capacity of RC 8000.

The manual contains in various sections descriptions of tableformats, messageformats etc. They are included for informative reasons only. A series of manuals is under preparation, describing in detail the conventions for the different functional layers in the network.

For the sake of brevity the RC 4000 computer has nearly not been mentioned in this manual. However in almost all cases it may do the same job as RC 8000. All communications software for RC 8000 is available in RC 4000 versions too.

2. Definitions and Concepts

This section gives some simple examples of communication systems in order to clarify the concept of computer network and related concepts.

As a first approximation a computer network is just a number of computers connected by communications lines permitting message exchange between the computers.

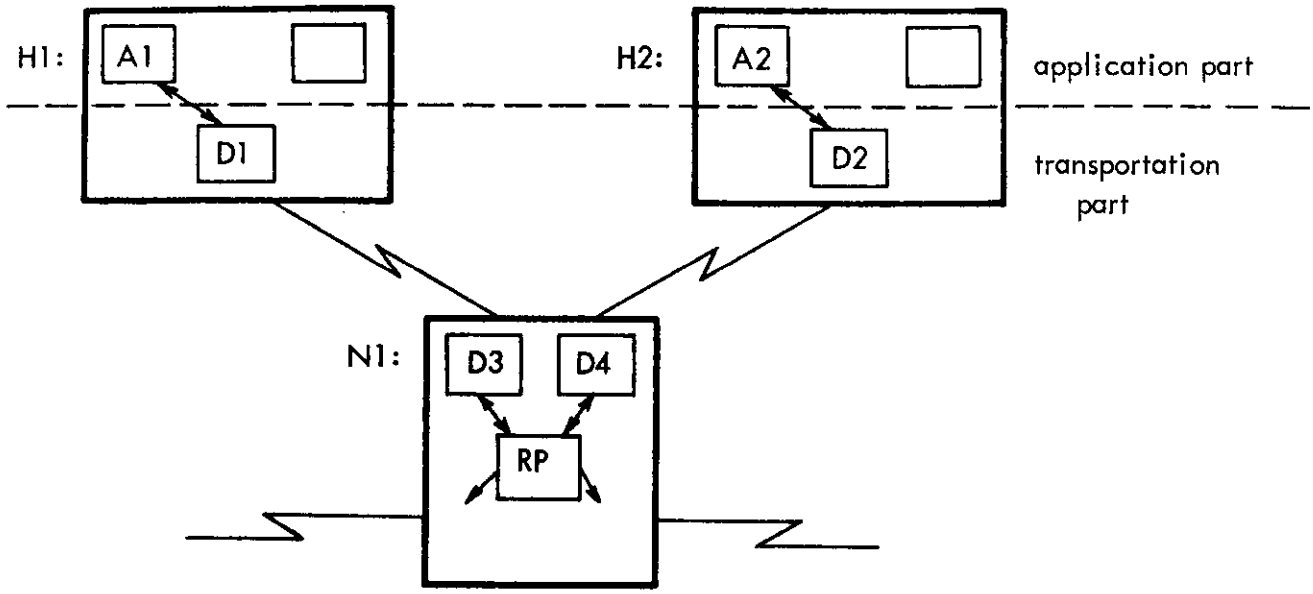
Normally one or several processes run in parallel in each computer in the network, and functionally one may divide the processes into two classes. The processes in the first class perform "normal data processing" tasks. This category comprises driver processes for peripheral devices as well as job processes executing user programs, and supervisor processes executing operating systems for the computer in question.

Each computer in the network has procedures for communication between (synchronization of) parallel processes in the computer. In a computer network the need arises for communication between processes running in different computers. This communication is done by means of a new category of processes the task of which is only (or mainly) to transport data travelling from a sender process to a receiver process.

The distinction explained above lead to a logical division of the network into a "transportation part" and an "application part".

The task of the transportation part is to move data around between processes in the application part, whose task is to process the data.

As an example a network is sketched consisting of two "job computers" H1 and H2, and a "communication computer" N1. A communication between the application processes A1 and A2 involves the communication drivers D1 - D4 as well as a "Network Contol Program" NCP which is responsible for the routing of data through the communication computer.



Ex. 2.1 Communication with remote peripheral device

H1: RC 4000
D1: FD, front end driver
H2: RC 3600 remote device controller
N1: RC 3600 network node
A1: BOSS 2 operating system
A2: Coroutine (device handler) for printer, terminal etc.

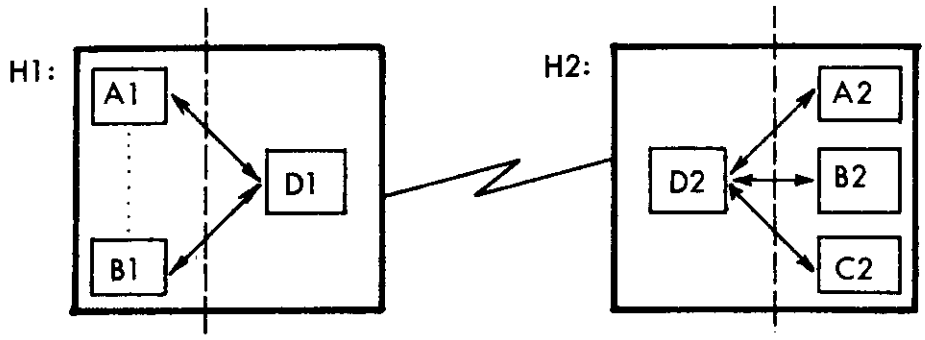
Ex 2.2 Move a disc file from one RC 4000 to another

H1: RC 4000
H2: RC 4000
N1: RC 3600 network node
A1: BOSS 2
A2: BOSS 2

A transportation network as shown above will be called a "data network". This manual describes such a network, the RCNET.

Nodes and hosts

A computer executing an application process is said to be a host for the process. In the examples 2.1 - 2.2, H1 and H2 are hosts for A1 and A2 respectively. Two hosts may communicate directly with each other:



application part datanetwork application part

Ex. 2.3 Direct host - host connection

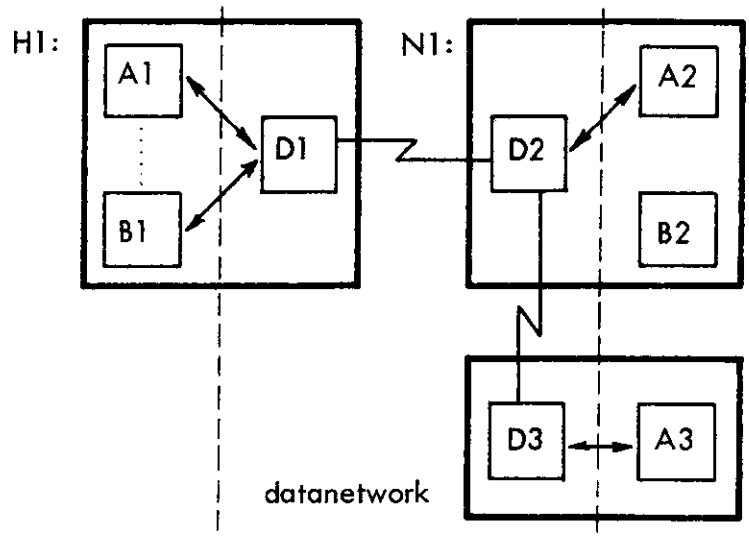
H1: RC 4000 with two online jobs.
H2: RC 3600 device controller with three terminals.

In the examples 2.1 - 2.2, H1 and H2 are connected through N1, and all data transfers between H1 and H2 have to pass through N1, which is called a node in the network.

A pure node is distinguished by the property that it transfers data between two application processes both running in other computers.

A pure host is distinguished by the property that it transfers data between two application processes one of which is running in the host itself.

The concepts of node and host are relative, as a computer may at the same time act as a node and as a host:

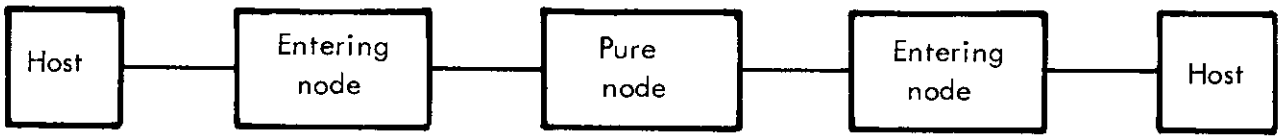


Ex. 2.4 N1 acts as node and as host

If a communication is going on between A1 and A2, N1 acts as host relative to this communication. At the same time B1 and A3 may be involved in a communication, and then N1 acts as node relative to that communication.

An entering node of the network is distinguished by the property that it connects a host to the network. It performs the conversion between messageformats used in the network and the formats recognized by the host.

The concepts may be visualized by this figure:

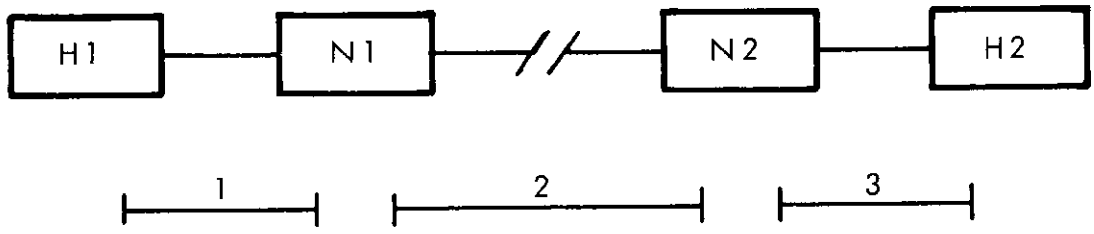


Link Concept

As mentioned above the main task of the data network is to transfer data quickly and reliably from a senderprocess to a receiverprocess. At a certain moment many portions of data may flow in the network originating at different senders and destined for different receivers. Thus every portion of data must contain information by means of which the network may direct it towards the correct receiver. A portion of data put into the network by a host is called a message.

A pair of processes involved in a communication is called a link, and the network keeps tables of the linka active for the moment. For each link is among other thongs kept information about the location of sender and receiver, the current state of the link, statistics and account information. Every communication between two processes must be started by one of the processes requesting a link to be created to the other process. Then the network will reserve and update the necessary table entries for identification of the link. A link is identified in the two hosts, as well as in the two entering nodes. It is identified in each place by a linknumber.

In order to make it possible to connect a variety of host computers to RCNET, a link between two application processes actually consists of three paths:



The entering nodes for the link are N1 and N2. Each message sent through the link must on its way between N1 and N2 carry information about destinationnode and the linknumber of the link in that node.

As the hosts may be of different types such as RC8000, IBM370, PDP11/45 etc., the identification of the link between H1 and N1, and between H2 and N2 will be hostdependent. Between N1 and N2 on the other hand, the link is identified in a standard way (section 7). Note also that there are special cases where two hosts are connected to the same node.

In ex. 2.4 the host H1 has two active links, A1 - A2 and B1 - A3. Each message from H1 to N1 must contain a linknumber by means of which the node N1 may direct the message to the correct receiver. Every message from N1 to H1 has correspondingly a linknumber associated with it so that the driver D1 may decide whether the message is destined for A1 or B1.

The formats chosen for messages and tables in RCNET limit the number of active links in a host or node. A host may have at most 1024 active links and a node may have at most 1024 active links.

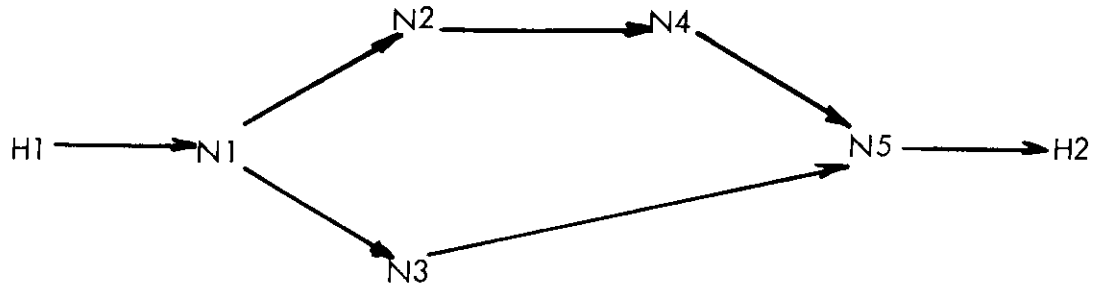
Messages and packets

Two processes connected by a link in RCNET communicate by exchanging blocks of information. Normally one of the processes sends a block and at a later time when the block has been processed another block is returned as an answer. Such blocks are called messages.

A message consists of a number of 8 bits bytes. The number of bytes is called the length of the message.

Average- and maximum length for messages over a link may vary greatly depending on the application. For TTY-terminals a message is typically shorter than 100 bytes. For VDU terminals on the other hand messagelength will frequently be about 2000 bytes.

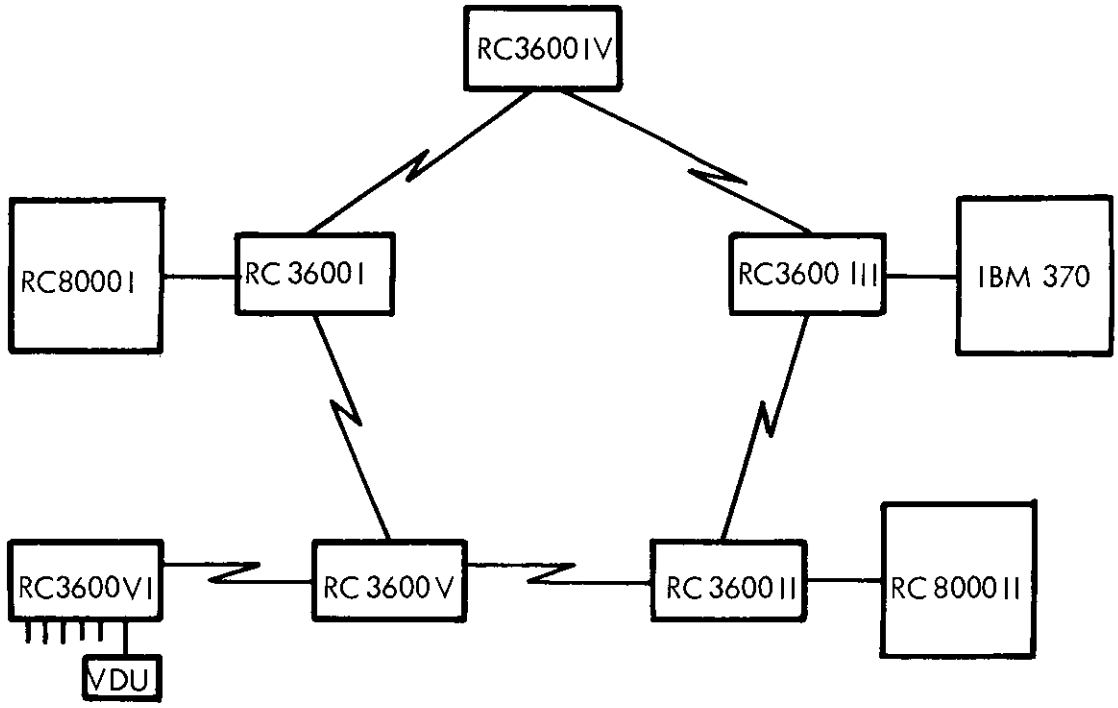
If a long message must pass several nodes on its way from sender to receiver the transmission delay may be unacceptable. The solution chosen in RCNET is to divide long messages into shorter packets in the entering node. The leaving node assemble the packets again before delivery to the receiving host.



In the network sketched above, the host H1 may for instance deliver a message of 1500 bytes to the node N1, destined for the host H2. N1 will divide the message into 7 packets each of 200 bytes and one of 100 bytes. The eight packets are sent independently to N5 and they may possibly use different routes. Probably they are also intermixed with packets originating from other messages. When all packets have reached N5 they are assembled to a message and delivered to H2. Up to this time N1 has kept a copy of the message which may be retransmitted in case of loss of one or more packets. Such a thing may for instance happen if a network node breaks down. When N5 has received all packets, an acknowledgement is returned to N1 telling it that the message may be released. This acknowledgement is called a PCA (packet control acknowledge).

3. A Network Example

This example will show how RCNET might be used as the basis for a communication system for a company. It is assumed that the data processing up to now has been batch processing on an IBM 370.



The IBM 370 is the central computer which among other tasks manages a central data base.

The RC 8000 computers perform special tasks such as production - and stock control of a plant or management of a local data base.

RC 3600 I and RC 3600 II:

Local device controllers for RC 8000 and network nodes.

RC 3600 III:

Communication front end for IBM 370.

RC 3600 IV and RC 3600 V:

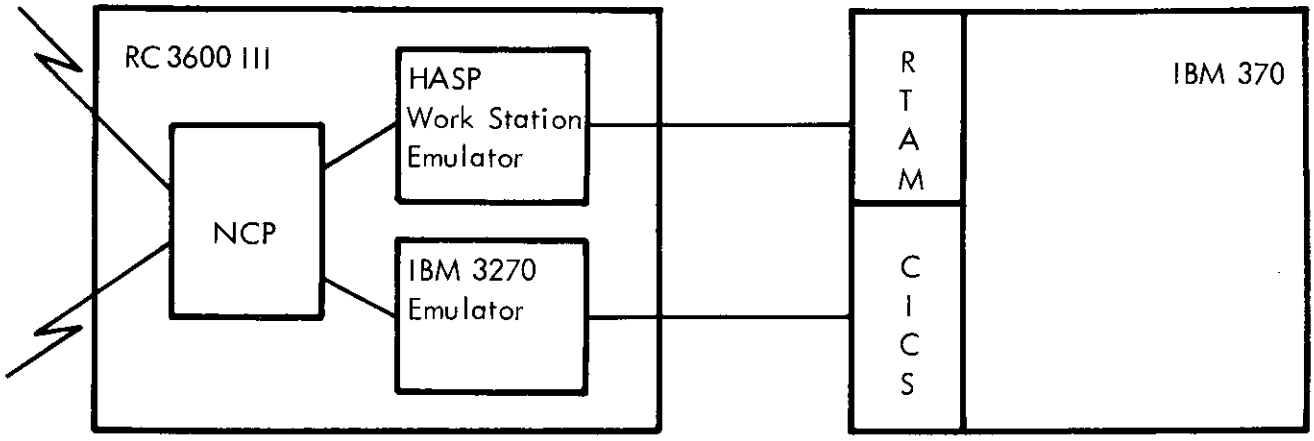
Network nodes and remote device controllers. They may act as terminal controllers, cluster controllers, RJE terminals, etc.

RC 3600 VI:

Remote device controller for VDU terminals.

The VDU terminals may for instance be used for inquiries. An inquiry may be sent to RC 8000 I which manages a local data base. If the inquiry can be answered by means of the data present here, the answer is sent to the terminal. If not, the inquiry is passed to the IBM 370. In this way the central computer is released from the load of simple inquiries which may be processed locally. Moreover the simple inquiries may be processed even in periods when the central computer is unavailable.

The RC 3600 III communication front end may be connected by more than one line, allowing use of different access methods in the IBM 370, e.g. RTAM and CICS.



Now the local data processing tasks of RC 8000 I may produce update transactions to the central data base. Those transactions may be sent as a batch using the HASP Work Station emulator, which may also be used for update transactions from IBM 370 to RC 8000 I, printing of lists on a printer connected to the remote device controller RC 3600 IV, etc. This device controller may also have attached to it a RC 3600 Data Entry System from which it may occasionally transfer data to the IBM 370 or a RC 8000.

4. Components of RCNET

This section will give a description of the hardware and software components of RCNET with most emphasis on a description of the software structure. The description will demonstrate a very flexible design allowing RCNET to be accommodated to a wide variety of applications. As an example one of the major design criteria has been that RCNET may be used as an interface between computers from different vendors such as IBM, Univac, etc., using existing communication software and access methods in these computers.

4.1 System Components

Virtually all equipment of the RC product line is from the start included in RCNET. Here a list is given of the major systems and their logical position in the network.

RC 8000

This is a computer with a large data processing capability. It has proven software for administrative data processing such as payroll administration, production and stock control. It has the powerful operating system BOSS 2 which is capable of batch - as well as online processing.

Also a general data base management system is on hand.

So RC 8000 is very suitable as central computer managing a central data base in small to medium sized communication systems, or as local computer in distributed systems.

Also it may serve as front end for a central computer in very large systems, releasing the main frame from routine tasks such as communication control, job scheduling, etc. The RC 4000 computer may serve the same purposes as RC 8000.

RC 3500

This is a communications computer which is specially suited to handle a number of fast communication lines. It is used as communication front end for RC 8000 if extremely large throughput is required.

RC 3600

This computer system is distinguished by a very large line of peripheral equipment of all sorts. Also a variety of communications equipment is on hand. Software packages are available for emulating most existing RJE terminals.

So the RC 3600 computer is used in RCNET as a network node when low or medium sized throughput is required. It is also used as a general purpose device controller making all the peripheral equipment of the RC 3600 line available for RCNET users.

As a local device controller for RC 8000 it handles magnetic tape stations, printers, readers, interactive terminals, etc. and acts as communications front end for RC 8000.

As a remote device controller it covers the functions normally performed by multiplexors, concentrators, cluster controllers, RJE terminals, etc.

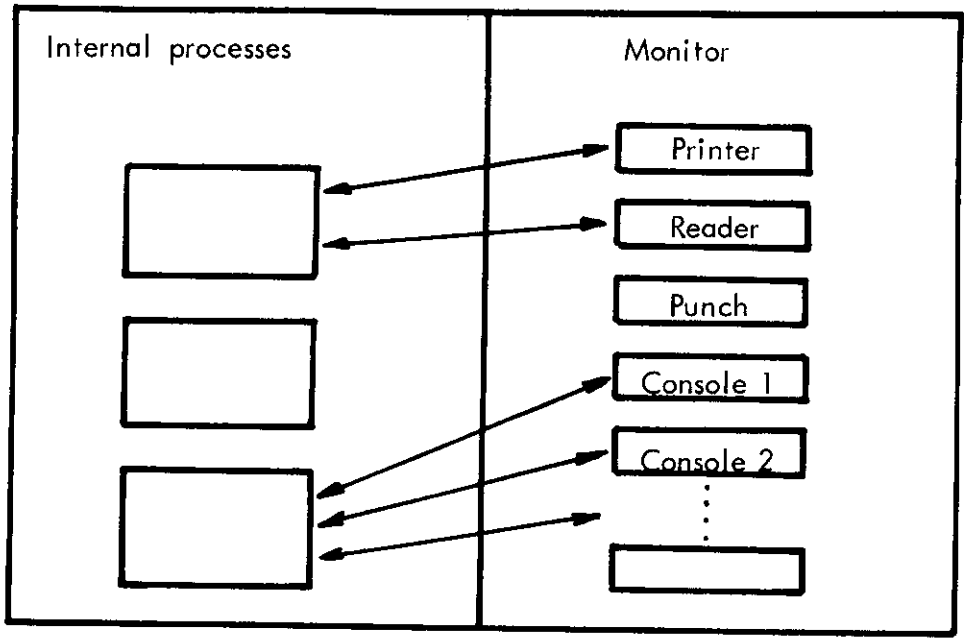
The RC 3600 Data Entry System may be used in connection with RCNET, transferring data through the network to a main frame.

4.2 RC 8000 Communications Software

First it is briefly sketched how local peripheral devices and backing storage files are represented and accessed by internal processes (user jobs or operation systems).

Each peripheral device or bs file is represented within the monitor by a named driver process. The internal process sends messages to the driver process, containing information about what to do, e.g. input or output operation, address of the data area within the sender process, etc. The driver process performs the desired operation on the device and returns an answer to the internal process, containing device status, number of bytes transferred, etc.

The scheme is sketched below, showing a process which communicates with printer and reader, and another process communicating with a number of console operators.



The description is somewhat simplified in case of jobs running under control of the BOSS 2 operating system, which in most cases spools the input or output for the job. This is, however, done behind the back of the job which may still believe that it has direct access to the device.

Remote Devices

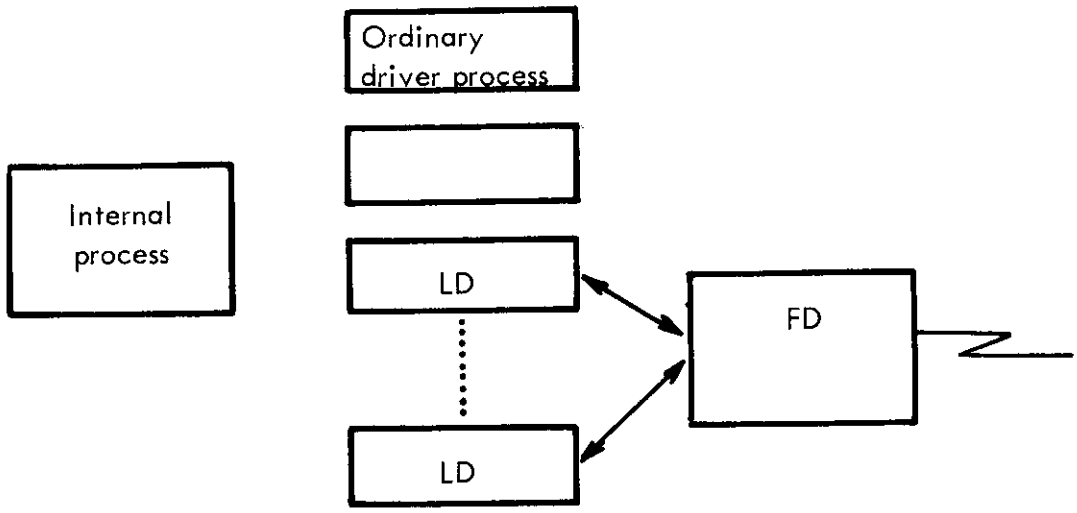
For the purpose of communication with remote devices through RCNET the monitor has been extended with a Front End Driver, FD, controlling a pool of so-called Link Driverprocesses, LD.

Initially all LD's are unassigned and do not accept messages from internal processes.

Now an internal process may request FD that a remote device, specified by its name in RCNET should be assigned to some LD or to a specific LD.

This will cause a link to be created through RCNET between the LD and the driver process in the device controller, controlling the device. The device may of course already be reserved by another computer, but then the request is queued in the device controller.

When the link has been created, the LD is initialized with information about the link and the kind of the device and from now on it accepts messages from the internal process just as an ordinary driver process. The LD, however, do not execute commands to the device such as an ordinary driver. Instead it passes the message and possibly the belonging data to the FD which sends it as a message through the link in RCNET.



It will be seen from the description above that as soon as the link has been created the remote device is accessed from the internal process by the same mechanisms used in communication with a local device.

When the process has finished the use of the device, e.g. printed a file, it will release the driver process. In case of a LD, this will cause the removal of the link in RCNET and the LD is set free so it is ready for connection to another remote device.

The initial request for creation of a link to the device may be done by means of a job control command (FP command in RC 4000 notion). So the program itself need not worry wether the device is local or remote, as they are accessed in the same way.

BOSS 2 Facilities

As mentioned before, BOSS 2 will normally spool input or output for a user job. For a remote device BOSS 2 will then use the scheme explained above.

Job control commands exist so that the job may request one or more remote files to be transferred to backing storage before the job is started. Moreover the job may during the run request one or more backing storage files to be transferred to one or more remote devices.

Finally it should be noted that the above mentioned software is running by now on the RC 4000 computer.

4.3 RC 3600 Device Controller Package

The RC 3600 software system consists of:

Multiprogramming monitor
Coroutine monitor
Basic operating system S
Driver programs
Application programs

The Device Controller Package includes a special application program, the Network Control Program, NCP.

This program makes the RC 3600 computer act as a local or remote device controller connected directly to RC 8000 or to a node in RCNET.

NCP performs the following tasks:

1. Controls the communication line to RC 8000 or to the network node according to the Line Control Protocol for that specific line (see section 6).

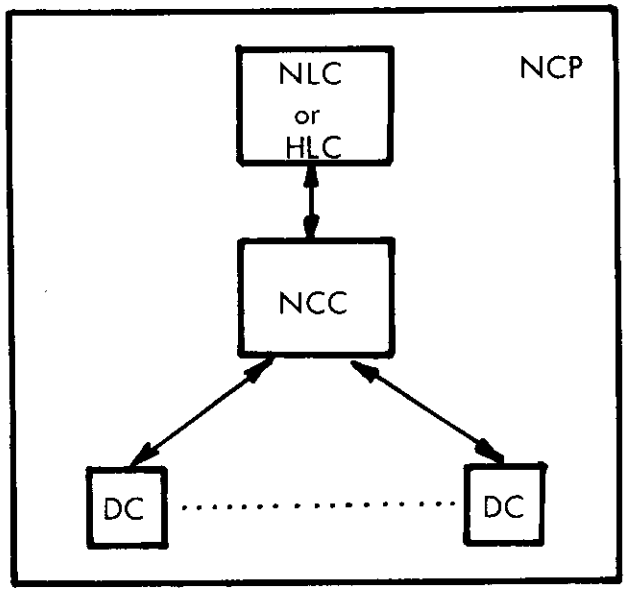
2. Controls a number of peripheral devices through the standard RC 3600 driver programs. Messages are received through links in RCNET, destined for the devices. NCP performs the desired operation on the device and sends an answer back through the link. The message traffic is handled according to the Message Control Protocol (see section 8).

3. Communicates with terminal operators about creation or removal of links, etc.

4. Handles requests from hosts about creation of links.

It is seen that NCP must execute a number of tasks in parallel. Each task is performed by a so-called coroutine. So NCP contains a coroutine for each device connected, and a line control coroutine for control of the communication line.

So the structure of NCP is as sketched below:



DC: Device Coroutine. Controls a single device according to the messages received through the link.

NCC: Network Control Coroutine. Controls creation and removal of links. Queues requests for devices from RC 8000. Controls the message flow between NLC (HLC) and the device coroutines by means of a link table (see section 7).

NLC: Node Line Coroutine.

HLC: Host Line Coroutine. Controls the communication line to the network node or the host computer according the the line protocol.

4.4 RC 3600 Network Node Package

The Network Node Package contains the Network Control Program, NCP, with Network Node Option.

A network node will normally act as a device controller as well. So NCP has the same duties as explained in section 4.3, but in addition to that is must:

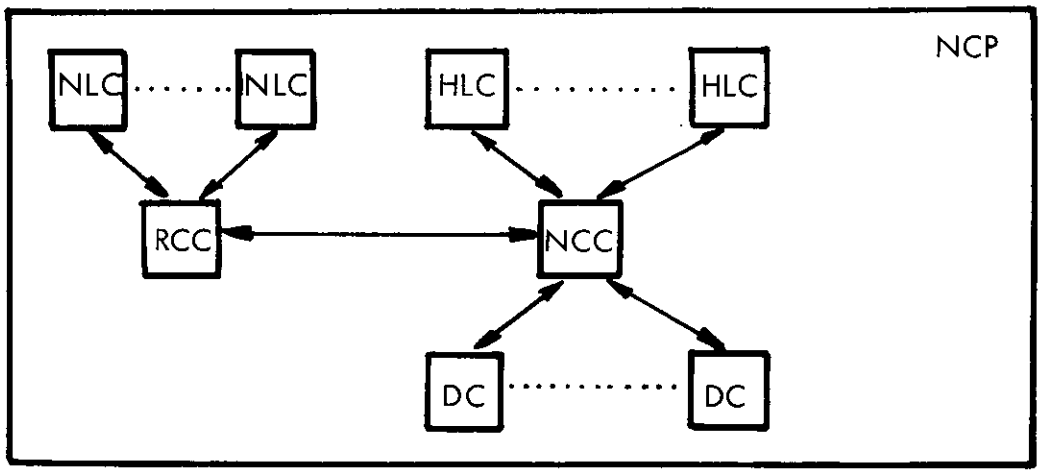
1. Receive packets destined for other network nodes and send them along according to routing tables (see section 7).

2. Update the routing tables according to received information about changing conditions in the network, such as congestion, failure of nodes or lines, etc.

3. Assemble packets to messages, pass them through to the destination host and send PCA (Packet Control Acknowledge) back to the sending node.

4. Split messages into packets and send them towards the destination node.

The structure of NCP in the Network Node Package is as shown below:



RCC: Routing Control Coroutine.
Receives packets from a NLC or NCC and passes them to another NLC or to NCC if they are destined for this node. Maintains the routing tables.

NCC: Will in this case perform the task of splitting massages into packets and assembling packets into messages. Moreover it will communicate with NCC's in other nodes according to the Packet Control Protocol (see section 7).

The Device Controller Package has been released, while the Network Node Package is still under preparation.

5. Protocol levels in RCNET

It is a well established rule in packet switching network technology that a number of logically independent control levels must be identified, e.g. communication line procedures should be independent of host-host message control procedures, etc.

This approach has at least two advantages. Firstly it will be easy to assure that failure of a system component has only isolated effects and leads at most to a performance degradation. Secondly it will be a lot easier to implement new strategies at different levels e.g. to change line control procedure if more efficient tools show up.

In RCNET the following three protocol levels are identified:

5.1 Line Control

On each physical communication line, a certain line protocol is used to assure errorfree transmission of data on the line without loss or duplication of messages. The protocol used between network nodes and between a network node and a RC 3600 devicecontroller is SDLC (ref. 3). On communication lines to computers or controllers from other vendors, the line control protocol will normally be the one used by the computer or controller, e.g. BSC for IBM equipment. The line control level adds Line Control Information (LCI) to each block before transmission, and strips it off before the block is passed to another level.

5.2 Packet Control

When the message enters the first node (entering node) in RCNET it is broken into a number of packets. Each packet is supplied with Packet Control Information (PCI). This information follows the packet until it arrives at the final node, where the packets are assembled into the original message.

5.3 Message Control

This is the end to end protocol, for instance the protocol between BOSS 2 in RC 8000 and a device coroutine in RC 3600 device controller. Each message contains Message Control Information (MCI) identifiing the type of the message.

Independent of these three protocol levels is the routing algorithm. The packet control information PCI contains information about the receiver node which the packet is destined for, but the determination of the actual path to follow is an independent task which in every node is done just before the packet is transmitted. The routing algorithm therefore may be changed according to arising needs.

The figure on the following page shows a data block on its way from BOSS 2 in RC 8000 to a printer on a remote RC 3600 device controller. It is seen how different program modules are responsible for the different control procedures and how control information is added and stripped off. For the sake of simplicity it is not shown how the message is split into packages on the packet control level.

The figure below shows in another way the logical aspects of the three protocol levels:

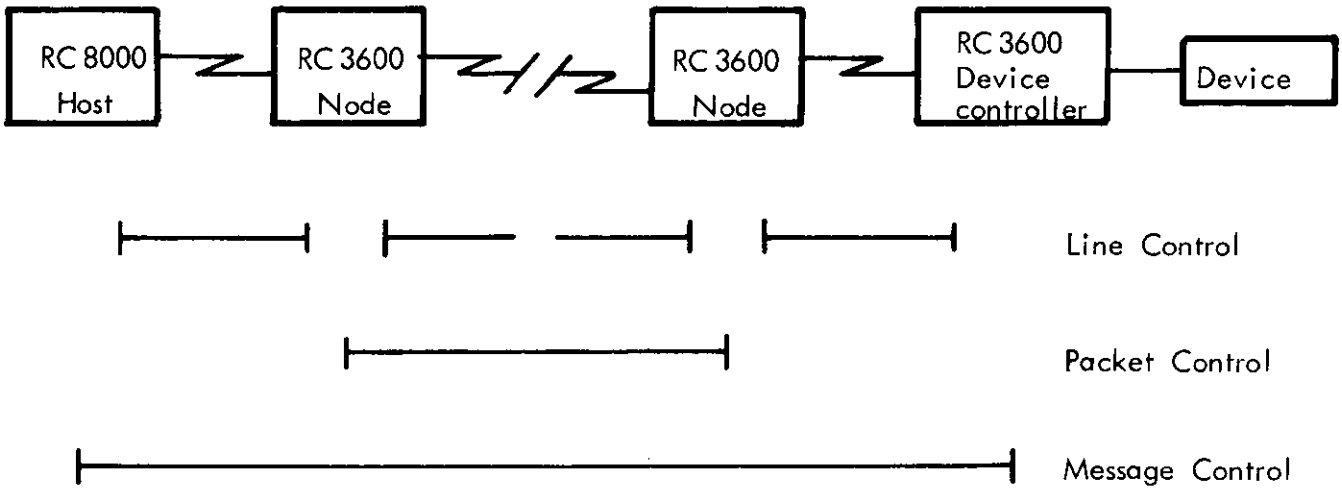


Fig. 5.1 Control levels in RCNET

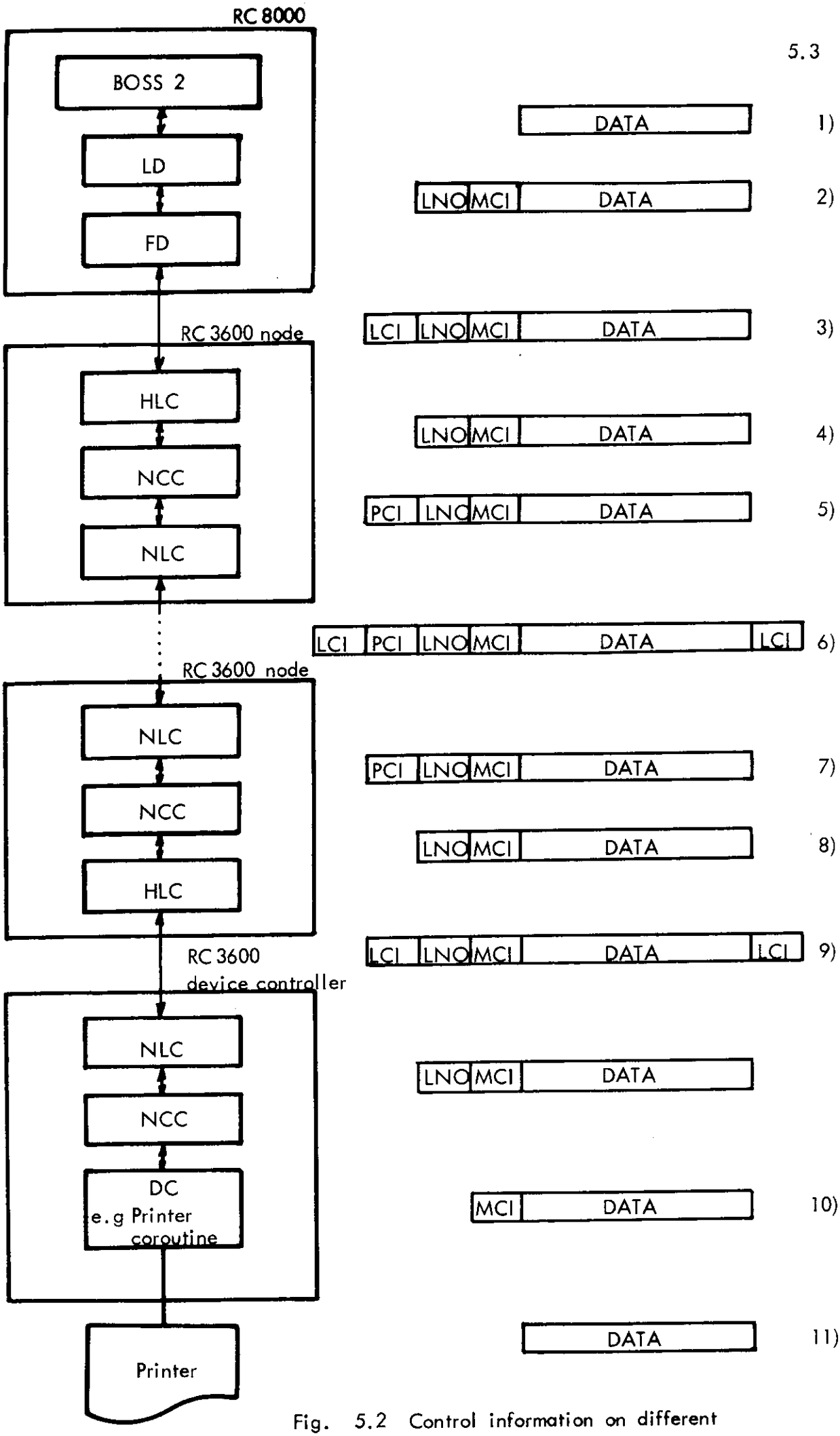


Fig. 5.2 Control information on different protocol levels.

In more detail the steps in the message flow are explained as follows:

1) BOSS 2 sends an outputmessage to the LD associated with the remote printer. The message points out the core area containing the data to be printed.

2) The LD adds Message Control Information, MCI and Linknumber, LNO before it passes the message to FD. MCI contains a functioncode, e.g. output, and some other fields explained in section 8.

3) The FD, which is responsible for line control adds LCI and transmits the block. It is received by the Host Line Coroutine HLC in the node adjecent to RC 8000.

4) HLC strips LCI off and passes the message to NCC.

5) NCC splits the message into packets (not shown) and adds Packet Control Information PCI to each packet. Then the packets are sent to the appropriate Node Line Coroutine NLC (actually to the Routing Control Coroutine RCC which chooses the NLC. See section 4.4).

6) NLC adds LCI and transmits the block. Here several nodes may lie in between.

7) Receiving NLC strips of LCI and passes the packet to NCC (actually to RCC which decides if the packet is destined for this node or should be passed through in the network).

8) NCC strips off PCI, assembles packets into a message (not shown) and sends the message to HLC. At the same time it sends a PCA back to the NCC at 4).

9) as 3) and 6).

10) NCC decides from the LNO which device coroutine will receive the data.

11) DC strips off MCI and outputs the data on the printer. When the data are printed an answer is sent back to BOSS 2. The answer will pass through the same control levels in the opposite direction.

6. Line Control

On each communication line or channel interface between two computers in the network, a line control protocol must be chosen. The purpose of the line control protocol is to assure efficient, error free transmission and to avoid loss or duplication of blocks.

The choice of line control protocol depends on:

1) Characteristics of the line.

   The following types of lines may be expected:

   Channel interface
   Telephone circuit
   Wide band circuit
   Satellite circuit

2) Available hardware.

   For instance BSC controllers or HDLC controllers on telephone circuits

3) Existing communication software.

   Computers or controllers from different vendors may impose certain line control procedures on the associated communication lines.

On the lines connecting RC equipment line control protocols have been defined as explained in the following sections:

6.1 RC 4000 - RC 3600/RC 3500

The connection is a synchronous communication line by modems or local cable. Speed range is 1200 bps to 1M bps.

The line control protocol has been defined with special regards to the characterictics of the SCC 401 communication controller used on RC 4000 in order to minimize overhead in the FD of RC 4000.

The line control protocol is described in ref. 1.

6.2 RC 8000 - RC 3600/RC 3500

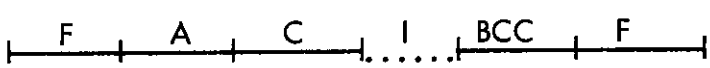
The connection is a parallel channel interface with a transfer rate of 600 K bytes/sec. The line control protocol has been defined with special regards to the characteristics of the FPA 801 front end adapter on RC 8000.

6.3 Network node - network node/network node - RC 3600 device controller

The "Synchronous Data Link Control", SDLC proposed by IBM has been chosen for those lines, as this was found to be the most versatile control protocol known at the moment.

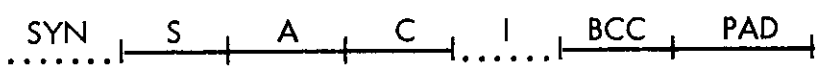
SDLC requires special hardware which can manage the HDLC procedure as described in ref. 2. As such hardware may not be available for the first version of RCNET, a modified SDLC procedure will be used in this version.

A block in SDLC has the following format:



(ref.3)

The modified protocol will use standard BSC hardware and the following block format:



The S-field contains 2^{15} + size, where size is the total length of A, C and I fields. When HDLC-hardware has been developed, the SDLC protocol may be introduced with only minor changes in line control coroutines.

7. Packet Control

The logical structure of RCNET is figured as a set of interconnected network nodes, each node having attached to it a number of hosts.

A host is just a computer executing one or more application processes. It may for instance be a large operating system in RC 8000 or a simple printer coroutine in a RC 3600 device controller. So an RC 8000 computer and a RC 3600 device controller attached to a network node are logically equivalent from the point of view of the network node.

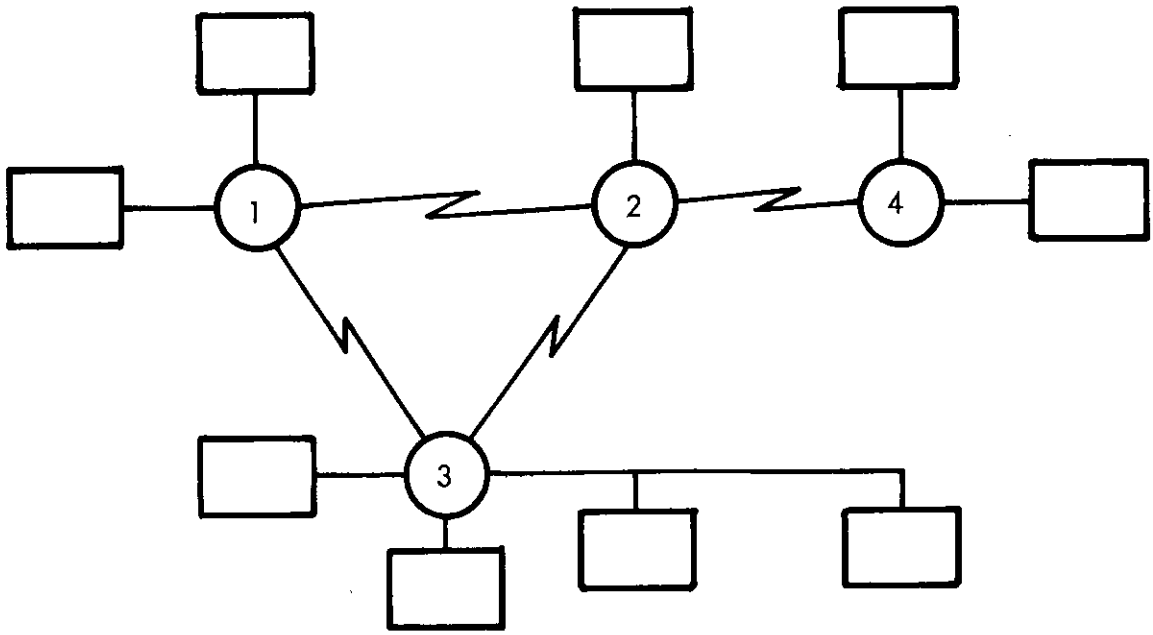
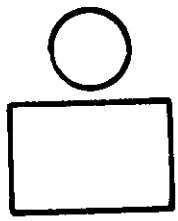


Fig. 7.1 Schematic network structure.



The hosts may be imagined to lie at the logical boundary of the network. A node may itself contain application processes (see fig. 1.4) so that it acts as a node as well as a host. From a logical point of view then, the host functions are pictured as a host attached to the node.

Each node is assigned number in the interval (0,127). Within each node, the hosts attached to the node have assigned numbers in the interval (1,15). The host number 0 is in each node reserved for the host function of the node itself.

Thus RCNET may consist of most 128 nodes, and each node may have attached to it at most 15 hosts.

Within each node the Network Control Program contains a Line Control Coroutine. LCC, for each communication line from the node.

An LCC for a line to another node is called a Node Line Coroutine, NLC.

An LCC for a line to a host is called a Host Line Coroutine, HLC.

All HLC's within a node are equivalent from a structural point of view even if the actual HLC's within a node may be quite different depending on the sort of communication line they control. This point may be clearer by an example.

The node no. 3 in fig. 7.1 may actually be a center of a configuration like this:

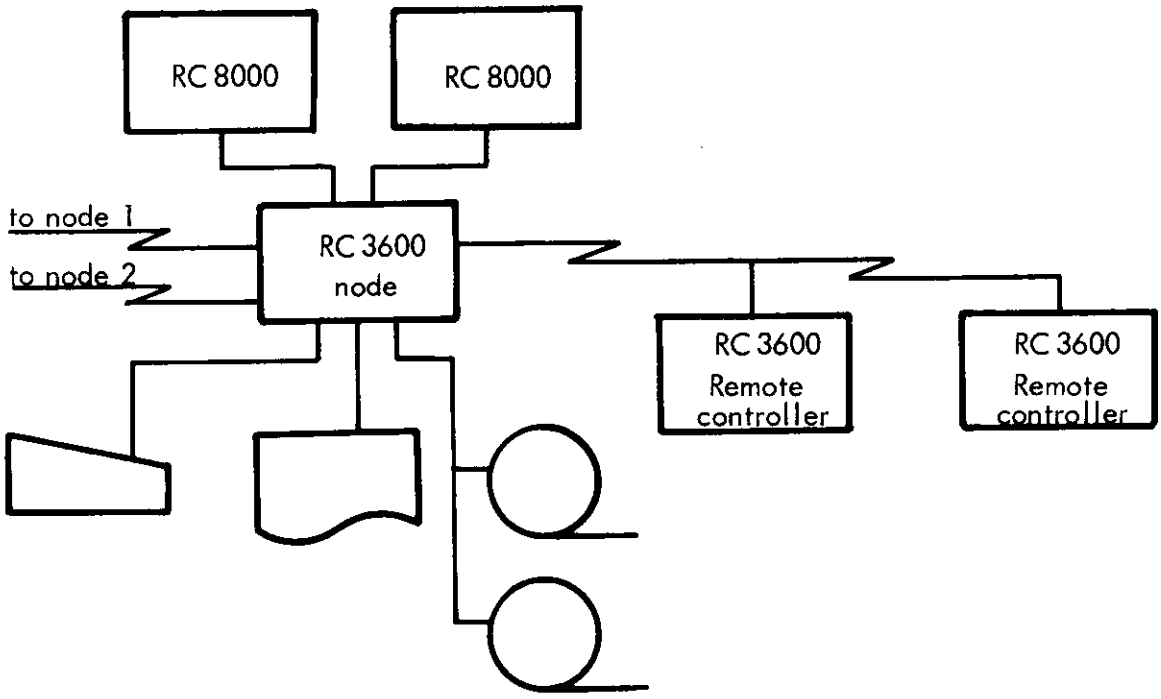


Fig. 7.2 The actual configuration of node no. 3 in fig. 7.1.
The RC 3600 node is local device controller for the two RC 8000.
The remote RC 3600's serve as terminal controllers or RJE terminals.

The RC 3600 node has two lines to other nodes, then two NLC's. It has three lines to hosts, then three HLC's. But two types of HLC are used, one for the RC 8000 connections and one for the RC 3600 connection.

NCP for this node has the following structure:

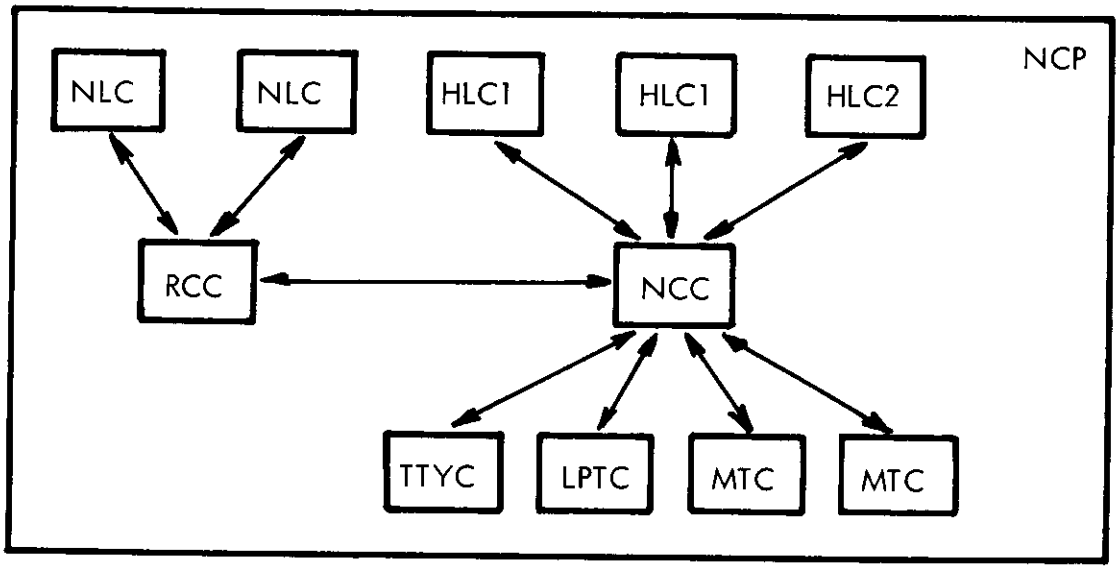


Fig. 7.3 NCP structure for the RC 3600 node in fig. 7.2.
HLC1: Host Line Coroutine to RC 8000.
HLC2: Host Line Coroutine to RC 3600 controllers.
TTYC, LPTC, MTC: Device coroutines for console, printer, magtape respectively.

Messages passing through node 3 may essentially use two different classes of paths within NCP.

The first class consists of messages from another node, not destined for node 3. Such data are received by a NLC and passed to RCC. It decides from the destination node field in the PCI of the message, that the message must be passed to a NLC, and the appropriate one is chosen by means of the routing tables.

The second class consists of messages originating at or destined for a host on node 3 (including messages to or from the device coroutines). All such messages are by fig. 7.3 seen to pass through NCC.

Messages belonging to the second class are obviously sent over links for which at least one of the endpoints is situated in a host on node 3. It could be a link from RC 8000 to the local printer or to a printer on one of the RC 3600 remote controllers. But it could also be a link from RC 8000 to a device on node 1 or from an RC 8000 on node 1 to a printer on an RC 3600 remote controller on node 3.

In the first case both ends of the link belong to a host on node 3. In the latter case one end of the link belongs to a host on node 3 while the other end belongs to a host on node 1.

Those links, for which at least one endpoint belongs to a host on node 3, are described by an entry in the linktable in this node.

A link for which the other end belongs to a host at another node is also described by an entry in that node.

7.1 Contents of link table entry

The size and meaning of the link table entry is as follows:

STATE 3 bits

Describes the state of the entry:

bit (0:1):
- 0 free
- 1 creating
- 2 removing
- 3 active

bit (2): spare

TYPE 1 bit

- 0 the link has only one end on this node.
- 1 the link has both ends on this node

For links of type 0:

NODE — 6 bits

Nodenumber of the node at the other end of the link.

NODELINKNO — 10 bits

The linknumber by which the other node identifies the link.

HOST — 5 bits

Hostnumber (inside this node) of the host to which the link belongs. Hostnumber 0 identifies the node itself when it acts as host.

HOSTLINKNO — 10 bits

The linknumber by which the host identifies the link. In a message to RC 8000 for instance, the hostlinkno identifies the LD which represents the link in RC 8000. In a message to a RC 3600 device controller, the hostlinkno is a device-number identifiing the device.

PRIORITY — 2 bits

Messages on links with higher priority are transmitted before messages on lower priority links, whenever a queue exists at a communication line. It may for instance be advisable to give terminal transaction priority over file transfer.

MESSAGENO — 3 bits

The next message sent to the other node of the link will have this number in the PCI of each packet. So the NCC at the other node may reassemble packets belonging to that message.

MY CRNO 3 bits

Creation number for this entry. Increased by one (module 8) each time a new link using this entry is created.

YOUR CRNO 3 bits

Creation number for the entry used for the link in the other end.

The creation numbers are used to catch packets which may have survived the removal of their corrosponding link and the creation of a new link, using the same entry in the link table. Each packet destined for this node must in its PCI have a creationnumber. If it is not equal to MY CRNO, the packet is discarded.

RMESS 16 bits

Head of the chain of messages being reassembled. When NCC receives the first packet of a message, bufferspace is reserved and chained to RMESS.

SMESS 16 bits

Head of the chain of messages sent to the other node, for which Packet Control Acknowledge (PCA) has not yet been received.

STATISTICS x bits

Statistics- and accountinformation for the link is accumulated here. It includes number of messages, number of characters, waiting times etc.

For links of type 1:

HOST1	5 bits. Host number of the host which requested the link to be created.
HOSTLINKNO1	10 bits. The linkno by which that host identifies the link.
HOST2	5 bits. Host number of the other host.
HOSTLINKNO2	10 bits. Linkno identifying the link that host
STATISTICS	x bits. As for type 0 links.

The procedure for creation and removal of links is described in section 10.

Link numbers are limited to 10 bits, thus the maximum number of links known to a node or a host is 1024. So the link table may at most contain 1024 entries. On the other hand it would in most cases be vastful to set aside core for that many entries. So each node has a parameter maxlink, the maximum number of links which may be active at the same time.

Normally the node will act as host for a set of peripheral devices and host number 0 is then the node itself.

The parameter maxdev for the node is the number of peripheral devices on the node. Each device is identified by a device number in the range (0, maxdev -1). The local devices have fixed assignments to link table entries, so they always occupy the first maxdev entries.

7.2 NCC Treatment of Messages from a Host

When NCC receives a message from a host (or a device coroutine) it uses the field LINKNO (see fig. 5.1, field LNO) to address the corresponding link table entry.

Then first the type of the link is inspected. For a link of type 1 the procedure is very simple. The host number of the other host is found from the entry. The corresponding hostlinkno is found as well and placed in the field LINKNO of the message. Then the message is passed over to the HLC which administrates the line to the host in question, or the message is passed to the appropriate device coroutine, if host number = 0.

For a link of type 0 the procedure is a little bit more complicated. First the MESSAGENO of the link table entry is attached to the message and MESSAGENO is increased. Then, if the message is longer than the packet length, it is split into packets. Then each packet is supplied with PCI and linkno LNO as described in section 7.3 (see also fig. 5.2). The packets are then passed to the RCC (see fig. 7.3) which determines where to send them. The message itself is put into the "chain of messages sent" in the link table entry, and a timer is started so that retransmission may be performed if PCA for the message is not received within the timeout period.

7.3 Contents of Packet Control Information

Before NCC sends a packet to RCC (see fig. 7.3) it attaches PCI to the packet. The size and meaning of the fields of PCI is as follows:

RECEIVER	6 bits. Node number of the destination node. Taken from link table entry, field NODE.
SENDER	6 bits. Node number of sending node.
PRIORITY	2 bits. Priority of the packet. Taken from link table entry, field PRIORITY.

CRNO	3 bits. Creation number of the link. Taken from link table entry, YOUR CRNO.
MESSAGENO	3 bits. Identification of the message to which the packet belongs. Taken from link table entry, field MESSAGENO.
PACKETNO	5 bits. Sequential number of the packet within the message.
NO OF PACKETS	5 bits. Number of packets in the message.
FUNCTION	1 bit. 0: the packet belongs to a normal user message 1: the packet belongs to a supervisory message.

The LINKNO field of the packet is taken from link table entry, field NODELINKNO.

The remaining part of the packet contains user information or supervisory information (if FUNCTION = 1).

7.4 NCC Treatment of Incoming Packets

Incoming packets from other nodes are, as mentioned before, inspected by RCC. If the field RECEIVER agrees with the node number of this node, then the packet is passed to NCC. The field LINKNO locates the corresponding link table entry. NCC will then use the field MESSAGENO to search for a partly assembled message in the chain RMESS. If no such message is found then the field NO OF PACKETS is used to reserve the buffer space necessary to assemble the total message. If buffer space is not available, the packet is discarded, relying on the timeout procedure at the sending node.

When buffer space has been reserved, a timer is started so as to control that all packets will arrive within a reasonable time. If timeout occurs, the buffer is released, again relying on the timeout procedure.

In a well-dimensioned network the above mentioned error conditions will be rare, so it is reasonable to use a timeout approach instead of some sort of negative adknowledgement in the Packet Control Protocol. Loss of packets, causing a timeout, may occur by two reasons:

1. An intermediate node falls out, so all packets waiting for transmission from that node are lost.

2. Congestion may occur in one or more intermediate nodes. The only way to avoid a deadlock situation in the whole network may be that some nodes drop some packets.

When NCC has received all packets belonging to a message it sends a PCA back to the sending node, and inspects the field FUNC of PCI. If the message is a supervisory one, the function is performed and an answer may be sent. Else the link table entry fields HOST and HOSTLINKNO are used. HOSTLINKNO is placed in the field LINKNO of the message. HOST is used to point out the HLC which must take over the message. If HOST = 0 the message is sent to a device coroutine instead.

7.5 Contents of Packet Control Acknowledgement

PCA is an ordinary packet sent through the link.

PRIORITY	3 (highest priority)
MESSAGENO	the message no. of the message for which acknowledgement is sent.
PACKETNO	0
NO OF PACKETS	1
FUNCTION	1

The data field consists of one byte. It is a function code for supervisory messages and the code in this case indicates that the message is a PCA.

8. Message Control

The set of network nodes with their connections constitute a transport subnet for for the hosts.

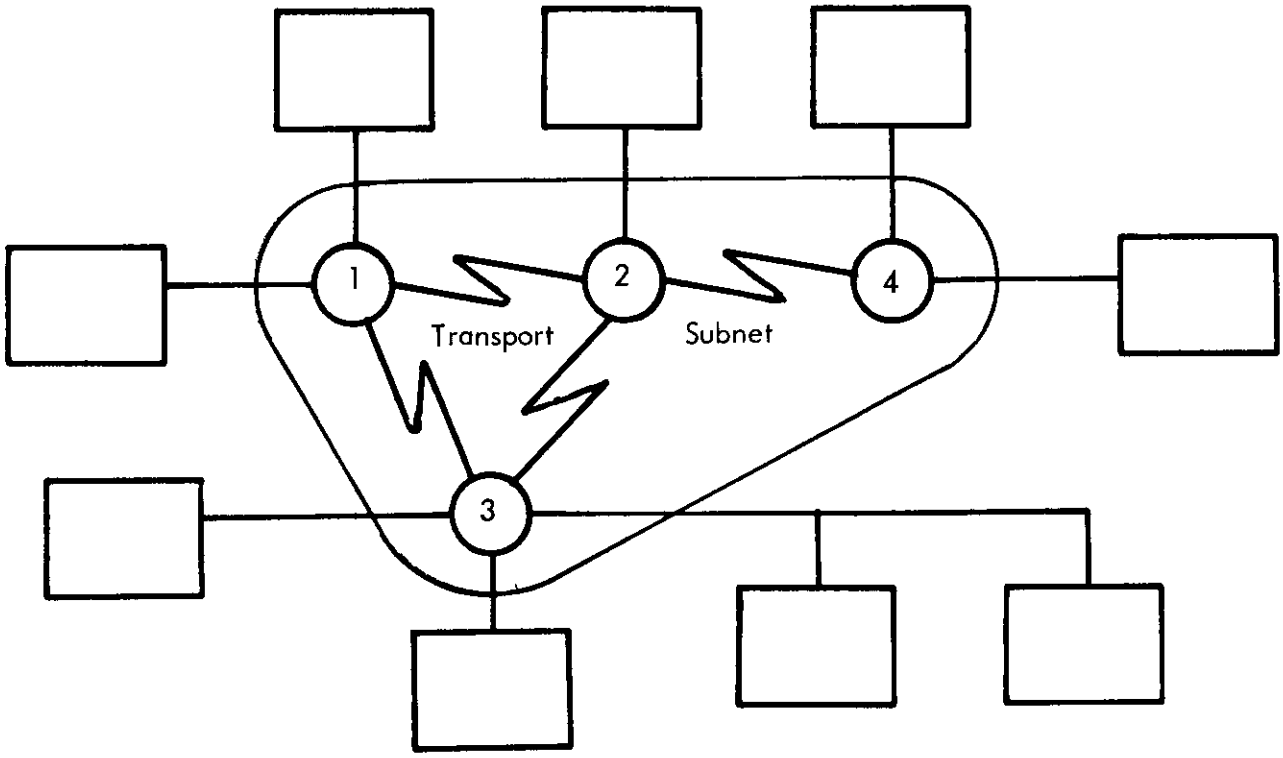


Fig. 8.1 The network in fig. 7.1 visualized as a transparent transport subnet connecting the hosts.

The hosts put messages into the subnet and receive messages from it, without having any knowledge about the actual subnet configuration, which may even change dynamically without any notice to the hosts.

So processes in the hosts may in principle communicate through the subnet by exchanging messages. However, in order that such a messageexchange between two processes can be usefull, the processes must agree about some principles for messageexchange, for instance agree about how to represent commandcodes, variable length datafields etc. This agreement between processes is called a Message Control Protocol. Such a protocol must be defined for each pair of processes connected by a link in RCNET.

The conventions explained so far impose very few restrictions on the Message Control Protocol. An important restriction is , that the subnet will not guarantee that the messages are delievered in sequence. So the Message Control Protocol uses a numbering scheme for messages assuring that the receiver processes the messages in correct sequence.

As another convention imposed by the subnet, the MCI must contain a function-code by means of which the NCC in a node may decide if the message is a supervisory one.

It is seen that several Message Control Protocols may be defined for use in RCNET. So far however, only a single one has been implemented. It will be described below.

8.1 Message Control Protocol for Device Control

This control protocol is developed for the purpose of connecting peripheral devices to RC 8000 by means of RCNET. It is briefly described in section 4.2.

As a protocol for control of peripheral devices, it shows a strict master-slave relationship. The master is the application process in RC 8000 and the slave is the devicecoroutine in the RC 3600 devicecontroller. The devicecoroutine will normally not perform any actions until it receives a message telling what to do. There are few exceptions to this rule. One is the action performed by a terminal coroutine if the operator strikes the attention key.

The description of the protocol will be carried out referring to this figure.

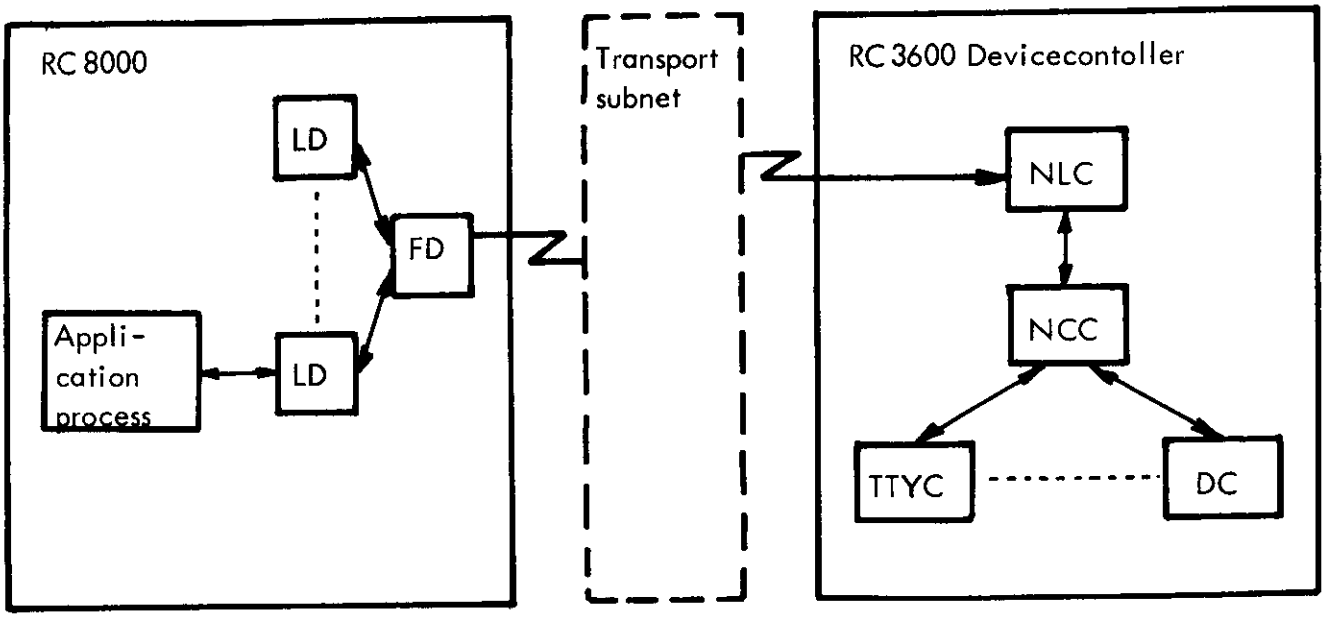


Fig. 8.2. Processes involved in message exchange.

A link between the RC 8000 application process and the typewriter coroutine TTYC in RC 3600 will be used as example. It is supposed that the link was created by the terminal operator, typing a select command (see section 11).

The purpose of the Message Control Protocol is that the RC 8000 process may treat the device just as if it was a local device of the same type.

So the format and contents of a message from application process to the LD, representing the device, is estabiliched by the conventions in RC 4000 Multi-programming System. The conventions for most devicetypes may be found in ref. 4.

The figure below shows how an output operation followed by an input operation is performed on a typewriter.

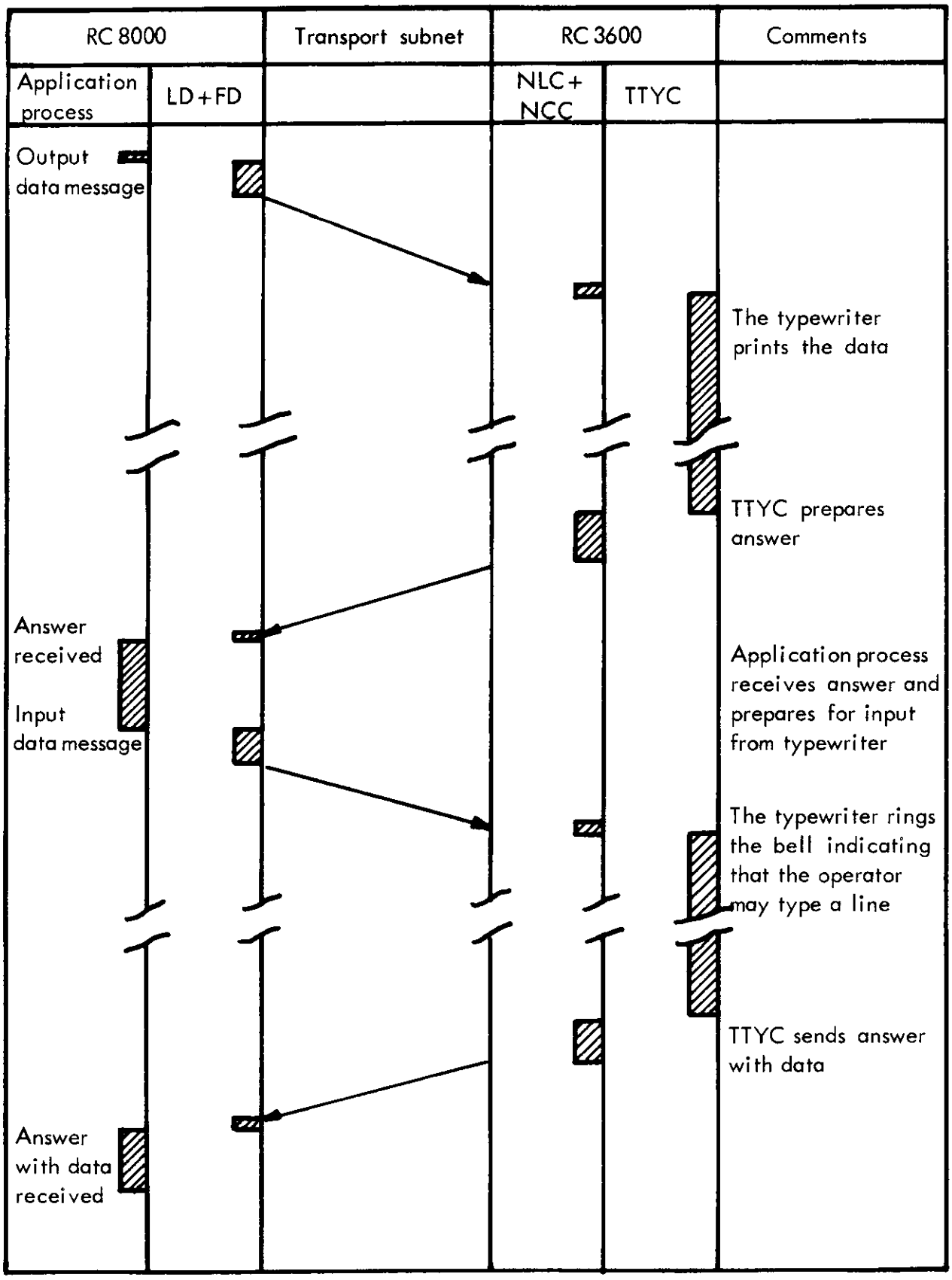


Fig. 8.3 Message exchange between application process (Boss 2) in RC 8000 and a typewritercoroutine in RC 3600.

▨ Processing- and queueing time

1. The application process outputs data to the typewriter by sending an outputmessage to the appropriate LD.
2. LD adds LNO and MCI to the data and FD puts a message consisting of LNO, MCI and data into the transport subnet through the link (cf fig. 5.1)
3. The message ends at the typewriter coroutine TTYC which activates the typewriter, such that the data are printed.
4. When the data have been printed (or an error on the device has occurred), TTYC sends an "answer output" back through the link.
5. When the application process receives the answer, the next operation may be performed. In this case an inputmessage is sent to the LD.
6. TTYC receives the inputmessage and rings the bell on the typewriter, indicating that the operator may type a line.
7. When the line has been typed, TTYC adds LNO and MCI to the data and sends an "answer input" message consisting of LNO, MCI and data through the link.
8. The data are received in the buffer of the application process, and it receives an answer from LD, so it can process the received typewriter line.

The above description shows the general scheme for message exchenge on a link.

The LD receives a message from an application process, converts it to the format used on the link and sends the message. The MCI of the message contains a functioncode defining the operation to be performed. When the devicecoroutine has performed the desired operation on the device, it sends a message back through the link, containing an answer to the application process. When the answer is received by RC 8000, the LD sends a normal answer to the application process.

With this scheme, as is seen from fig. 8.3, the device is idle from the time the answer is sent, until the next message is received by the devicecoroutine. This will cause no problems for terminal traffic, as long as transmission through the transport subnet is reasonably fast. For file transfer however, for instance to a printer, a better approach must be used, such that the printing and transmission may overlap.

If the application process uses a double buffer for output, it may send the next outputmessage to the LD without awaiting the answer of the previous one. The LD will send the output message at once, and so the desired result is obtained. As each message from LD has a message sequence number in its MCI, the printercoroutine may print the data in the correct sequence, even if two messages are occasionally delivered out of sequence by the transport subnet.

8.2 Contents of message and answers in the protocol.

The Message Control Information MCI contains 5 fields f1-f5 with length:

f1	6 bits
f2	4 bits
f3	12 bits
f4	14 bits
f5	5 bits

For input- or outputmessages and their answers the fields have following meaning:

f1:	FUNCTION	Input:	44
		Answer input:	46/47
		Output:	49
		Answer output:	50
f2:	STATE	In all messages to the device, state tells whether the device must be initialized.	
	RESULT	In all answers, RESULT gives a result of the operation, for instance if the message was unintelligible or the device was disconnected.	
f3:	MODE	In a message the mode specifies parity, codeconversion, density etc.	
	STATUS	In an answer the status contains devicestatus at completion of the operation.	
f4:	SIZE	Input:	The maximum number of characters to be input.
		Answer input:	The actual number of characters input.
		Output:	The number of characters to be output.
		Answer output:	The actual number of characters output.
f5:	MESSAGENO	Sequencenumber used for correct sequencing of messages. Is returned with the answer, identifiing the corresponding message.	

An even function indicates that the datafield of the message is empty.

In addition to input and output messages a number of other messages are defined. In particular a number of supervisory messages are defined:

CREATE	Create a link
REMOVE	Remove a link
RELEASE	Release the device
DIRECT MESSAGE	Message not sent through a link. MCI then contains the network address of the destination host.

Supervisory messages are distinguished by having a function code in the range (0,31). Normal messages have function codes in the range (32,63).

8.3. Needs for Other Types of Message Control Protocol

As mentioned, the protocol described is a master-slave protocol, particularly developed for device control. It does not easily lend itself to more general communications, e.g. between two application processes running in two different RC 8000's. A communication of this sort might be useful in a distributed data-processing system with co-operating jobs running in different hosts. A job in one host may for instance need one or more records from a data base administrated by a special data-base processor.

9. Routing Principles

This section describes the principles used for the routing of packets within the transport subnet (Fig. 8.1.)

The purpose of the routing algorithm is to choose optimal or near-optimal paths for the packets traversing the transport subnet. This task is performed by the routing control coroutine RCC in each node. The information used is the receiver field in each packet, containing the node number of the destination node and some kind of routing table by means of which the RCC may choose a communication line (port) for transmission of the packet.

The description will be carried out referring to the following network with five nodes:

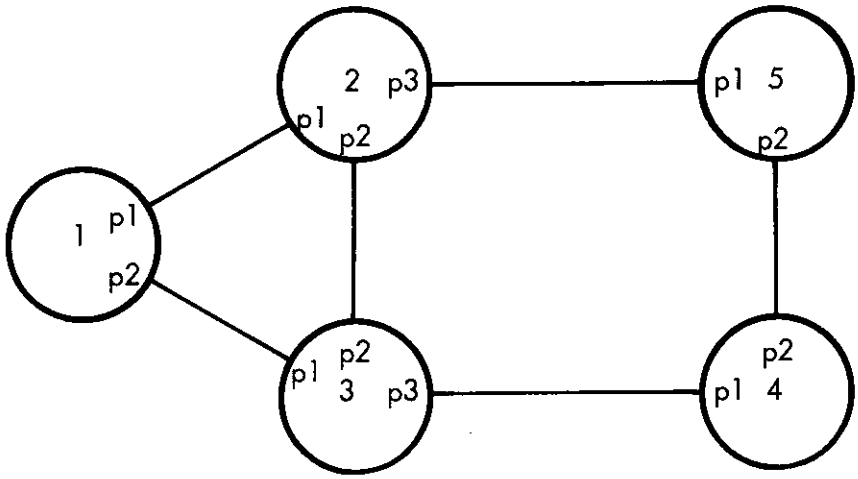


Fig. 9.1.

Three different routing principles will be briefly described. They are:

1. Fixed Routing
2. Shortest Path
3. Delay Estimate

In the following N designates the total number of nodes in the network.

9.1. Fixed Routing

With this method, each node has a fixed table with N entries. Entry n contains the identification of the port to be used for packets destined for node number n.

The routing tables for the nodes in Fig. 9.1. may then look like this:

Table in / Entry	node 1	node 2	node 3	node 4	node 5
1	0	p1	p1	p1	p1
2	p1	0	p2	p2	p1
3	p2	p2	0	p1	p2
4	p2	p3	p3	0	p2
5	p1	p3	p3	p2	0

This method has the advantage of being very simple. As long as all network nodes are functioning, it will work well.

However, a rather undesirable situation may arise, if one of the network nodes breaks down. If e.g. node 3 fails, packets may by no means be sent from node 1 to node 4, as the routing table specifies that all packets from node 1 to node 4 must pass through node 3. So node 1 is disconnected from node 4 even if a physical connection remains.

It is seen that the situation could be repaired if entry 4 of the routing table in node 1 could be temporarily changed from p2 to p1.

9.2. Shortest Path Routing

Fortunately a very simple approach can be used, by means of which the routing tables may change dynamically because of changing conditions in the network.

With fixed routing the routing table in each node is a vector of N elements, where N is the number of nodes in the network.

With Shortest Path Routing, the routing table in each node is a NxM matrix, where M is the number of ports in that particular node.

Row n in the matrix contains the estimated path lengths to node n using the different ports.

The routing matrices for node 1 and node 2 when the whole network is functioning are shown as an example. The "choice" column shows the best choices of port at the moment, and the "path vector" column shows the estimated length of the paths when the best choice is used.

	p1	p2	choice	path vector
1	2	2	p1	0
2	1	2	p1	1
3	2	1	p2	1
4	3	2	p2	2
5	2	3	p1	2

Fig. 9.2. Routing matrix for node 1 in Fig 9.1.

	p1	p2	p3	choice	path vector
1	1	2	3	p1	1
2	2	2	2	p1	0
3	2	1	3	p2	1
4	3	2	2	p3	2
5	3	3	1	p3	1

Fig. 9.3. Routing matrix for node 2 in Fig. 9.1.

At regular intervals each node sends its path vector to all its neighbour nodes. Node 2 will, for instance, send to nodes 1, 3, and 5, and receive path vectors from the same nodes.

When a path vector is received, all entries are increased by one, and the resulting vector is inserted in the routing matrix in the column corresponding to the port which received the vector. The best choice and path vector are then recalculated.

When a node finds a port closed because of hardware failure in the port or the communication line or because the other node fails to respond, all entries in the column corresponding to the port are set to N+1, where N is the total number of nodes in the network and the path vector is recalculated again.

Imagine now, as an example, that port 3 of node 2 fails. Then the routing matrix in node 2 will be updated and have the following contents:

	p1	p2	p3	choice	path vector
1	1	2	6	p1	1
2	2	2	6	p1	0
3	2	1	6	p2	1
4	3	2	6	p2	2
5	3	3	6	p1	3

Fig. 9.4. Routing matrix for node 2 when port 3 is closed

The path vector is now sent to node 1 and node 3. After updating the routing matrix in node 1 looks like this:

	p1	p2	choice	path vector
1	2	2	p1	0
2	1	2	p1	1
3	2	1	p2	1
4	3	2	p2	2
5	4	3	p2	3

Fig. 9.5. Routing matrix for node 1 when path vector from node 2 has been received

A comparison with Fig. 9.2. shows that the choice has changed as far as node 5 is concerned. Thus the network accommodates to changing conditions by means of the algorithm indicated.

9.3. Advantages of Dynamic Routing

A couple of other advantages of the Shortest Path algorithm should be noted.

Firstly, each node will easily discover it if it cannot reach another node. This will be the case if, and only if, the corresponding entry in the path vector exceeds N, the number of nodes in the network.

If a node discovers that another node which has been reachable becomes unreachable, it will inspect its link table to see if the disconnected node is involved in any links. If so, the host may be informed that the link is disconnected.

Another advantage concerns more permanent changes in network configuration by introduction of new nodes or new communication lines. With Fixed Routing such a change will require that new reuting tables be introduced in the programs of all network nodes.

With Dynamic Routing the problem is almost non-existing. The only thing to be done is to estimate the number of new nodes to be introduced within a year or two. Also, for each node it must be estimated how many ports may be necessary in the future. Thus the routing matrices may from the start be dimensioned according to the future demands.

Each node will now be on the watch for the occurrence of connections to the closed ports. Thus a new node may be introduced simply by connecting it to one or more existing nodes. The connections having been opened, the whole network will be aware of the new node after a while.

New connections between existing nodes may also be introduced, and the network will quickly accommodate to the new configuration. Some nodes may be equipped with ports which may be called through switched telephone lines.

Note in this connection that a node is always identified to its neighbours by the zero in the corresponding entry of the path vector that it sends.

9.4. Delay Estimate

The Shortest Path method is a rather simple one and will be used in the first version of RCNET. It will work excellently as long as the network is not heavily loaded. However, if the load exceeds a certain limit, the method displays some drawbacks.

Firstly, reference to Fig. 9.1. will show that if there is heavy traffic between node 1 and node 5, all this traffic will pass through node 2 even if node 3 is at the same time nearly idle. Of course the ideal situation would be a smoothing of traffic over several routes.

Moreover, in the case of heavy loads, congestion in a node or a cluster of nodes will be more likely. As described in section 7, measures have been taken to prevent deadlocks caused by congestion, but those measures may cause a degraded throughput if congestion occurs frequently.

To overcome such problems a more sophisticated routing method must be used.

With the Shortest Path method each node tries to estimate the length of the shortest path to every other node. Now, what we are really interested in minimizing is not the path length to another node but the time needed for a packet to reach that node. If all communication lines between nodes have the same transmission speed, and if there are no queues at the ports of intermediate nodes, transmission delay is proportional to path length.

With a heavily-loaded network containing different line speeds better approximation to real transmission delays are available. Still the method of routing matrix and exchange of vectors is used, but the entries are now estimates of transmission delays. This estimate is taken as a weighted mean of queue lengths and line capacities (inverse ratio of bit rate on the port).

This routing method is very close to the one used in the ARPA-network. It will be introduced into RCNET when load considerations are in favour of such a solution.

10. Creation and Removal of Links

This section describes how links are created and removed. Some related subjects, such as methods of identification of hosts and processes, are discussed besides.

Before two processes may communicate they must have been connected by a link.

A link is in principle created by one host sending a "create" message containing an identification of the other host. At the most primitive level the identification is simply the physical network address, which consists of the node number and the host number within the node.

Thus a network address is an 11 bit integer with the format

node number * 16 + host number

Later on it will be described how a more user-oriented identification may be introduced.

In the case of a link between RC 8000 and a device in an RC 3600 device controller it might seem natural for the "create" message to be sent by RC 8000. It has been decided, however,that the "create" message will always be sent by the RC 3600, and there are two main reasons for this decision.

Firstly, for interactive terminals the initiative comes from the human operator, who chooses the host with which he wants to communicate. In this case it is therefor natural that the device controller should have an initial dialogue with the operator, who gives an identification of the desired host. The device controller then converts the identification to a network address and sends a "create" message to the host at this address. After a while it receives an answer with a result showing whether the creation was successful.

It might be unsuccessful for a variety of reasons:

1. The host was disconnected or out of operation
2. The host had at the moment no resources (LD's) for the link
3. An intermediate node had no resources (link-table entries) for the link
4. The host was unknown to the other node

For other kinds of devices, such as printers, readers, etc., the initiative will normally come from an application process wanting access to the device. It might thus in this case be natural for the "create" message to be sent by RC 8000. Now, the device might already have been reserved by another host at the time when the "create" message arrived, and in order to resolve such conflicts in a reasonable way it would be necessary to introduce a central network operating system to handle all requests for devices and queue requests that could not not be honoured at the moment.

Now, for the sake of reliability it would be unacceptable that all device operations should rely on a single host, which might of course break down. The network must therefore be able to work without a central operating system.

For this reason a solution has been adopted where the application process that wants a certain device sends a so-called "lookup" message to the device controller. If the device is not engaged, the device controller will send a "create" message to the sending host, and when the answer arrives, an answer is returned for the "lookup" message. If the device is engaged, the "lookup" message is queued in the device controller, and when the device becomes unengaged later on, the waiting "lookup"s may then be served on a first come-first served basis or according to some other strategy.

On the other hand it may in some cases be desirable to use a network operating system. Conventions allowing this have therefore been set.

10.1. Contents of "create" Message

The fields of a "create" message contain the following information (cf. section 8.2.):

f1 function code = o for create

f2 not used

f3 network address of receiving host

f4 devicekind, blocksize
devicekind is a code defining the type of the device
blocksize states the maximum message length accepted

f5 no of buffers
the number of buffers set aside in the device controller for the device

The answer contains the following:

f1 function code = 2 for answer create

f2 result
- 0 ok
- 1 already created (programming error!)
- 2 - -
- 3 no resources
- 4 host unknown

f3 RC 8000 deviceno of the LD connected to the link

f4 node linkno
the linkno of the link in the node adjacent to the device controller. Must be used as LNO in all messages and answers from the device controller.

f5 not used

10.2. "lookup" and "include" Messages

As mentioned before, the "lookup" message is a request for a specified device (or a device of a specified kind) sent from an application process in RC 8000. If the device is not free, the requests will be queued, and they will normally be served on a first come-first served basis.

In certain cases one or more hosts can have a higher priority than other hosts in a network. Two cases are dealt with here.

1. If the RC 3600 is local device controller to an RC 8000, this has the highest priority with the device, as it will normally want to treat it as a local device always ready for operation.

2. A network operating system must be able to schedule the use of devices between all the hosts in the network.

For these two purposes the so-called "include" messages are defined. They ask the device controller to include a specified device in the configuration of a specified host.

Device names are used for the identification of devices within a device controller. They are local entities so that a global identification of a device consists of the device name together with the network address of the device controller.

"lookup" and "include" messages are not sent through a link but addressed direct to the device controller. The function code 5 in the messages indicates that the messages do not belong to a link.