RcPAX Network System

The Data Transmission System

General Description

Keywords:
RcPAX, RC5000 MegaSwitch, RC5000 Network Switch, ROUTER, LIC, DCE, STE, IDTE, NCP, DIS, ATS, SL, TL, POOLMAN, X.25, X.75.

Abstract:
This document provides an introduction to the Data Transmission System of the RcPAX Network System. First an overview of the Data Transmission System, identifying the different Software Entities is given. Then the ROUTER, LIC, DCE , STE and IDTE Software Entities are described in more details.

Date:
October 1991

Author:
Inger Bohlbro, Ole Bromose, Kristian Gregersen, Jørgen Katborg og Valther Rasmussen

PN: 99001391

Table of Contents

1. Introduction 1

2. System Overview 3

3. Software Components 9
   3. 1. Datagram Service 10
   3. 2. Connection Service 12
   3. 3. Access Services 13
   3. 4. Associated Services 15

4. ROUTER 21
   4. 1. Routing Algorithm 22
   4. 2. Interface 23
   4. 3. Protocols 24
   4. 4. Routing Service 26
      4. 4. 1. Routing Hierarchy 26
      4. 4. 2. Routing Tables 28
      4. 4. 3. Congestion Control 31
      4. 4. 4. Estimating Costs 32
      4. 4. 5. Resource Allocation 33
   4. 5. Additional Services 33
      4. 5. 1. Network Clock 33
      4. 5. 2. Cyclic Redundancy Code 34
      4. 5. 3. Performance Management 34
      4. 5. 4. Line Configuration 35
      4. 5. 5. Broadcast 35
      4. 5. 6. Router-Router Packets 36

5. The Logical Internal Channel (LIC) 37
   5. 1. The LIC Services 38
   5. 2. The LIC Service Interface 42
      5. 2. 1. LIC establishment 44
      5. 2. 2. Lic removal 45
      5. 2. 3. Lic reset 46
      5. 2. 4. Packet transfer 47
      5. 2. 5. Increase credit 48
      5. 2. 6. Message transfer 49
      5. 2. 7. Datagram transfer 50
      5. 2. 8. Buffer Lack 50
      5. 2. 9. Set input buffer threshold 51

5. 3. The LIC protocol 51
5. 3. 1. LIC header 52
5. 3. 2. LIC protocol operation 54
5. 4. LIC Functions 57
5. 4. 1. Retransmission 57
5. 4. 2. Resequencing 58
5. 4. 3. PDU integrity 59
5. 4. 4. Heartbeat 59
5. 4. 5. Cancel resequence queue 60

6. DCE and STE 61
6. 1. General overview 61
6. 2. Services offered by the DCE/STE 63
6. 2. 1. Increased line utilization facility 63
6. 2. 2. 80/84 interworking 64
6. 2. 3. RR-Timer 65
6. 2. 4. Address Translation Service 65
6. 2. 5. NUI validation 65
6. 2. 6. Call/Data-timers 66
6. 2. 7. X.2 Facilities supported by the DCE 66
6. 2. 8. X.75 Network utilities supported by the STE 68
6. 3. Services used by the DCE/STE 69
6. 4. External Interfaces 69
6. 5. Functions 70
6. 5. 1. Facilities/utilities in call setup and clear 70
6. 5. 2. Buffer flow 71
6. 5. 3. Flow control 72
6. 5. 4. The M-bit concatenation mechanism 75

7. The Internal DTE (IDTE) 77
7. 1. The IDTE Services 81
7. 2. The IDTE service interface 87
7. 2. 1. Ivc Establishment 88
7. 2. 2. Ivc removal 89
7. 2. 3. Ivc Reset 91
7. 2. 4. Packet Transfer 93
7. 2. 5. Interrupt Transfer 93
7. 2. 6. Datagram Transfer 94
7. 2. 7. Dedicated receive buffer too small 95
7. 2. 8. Change Receive Mode 95

8. References 97

1. Introduction

The RcPAX data communication system provides efficient data communication services for both public and private data communication networks. An overview of the RcPAX system, outlining both hardware and software components, can be found in "RcPAX, An Introduction", (ref. 1).

This document provides a description of the Data Transmission System part of the RcPAX system focusing on the software components.

The description is in two parts:

Part I: Provides an overview of the Data Transmission System, its services and the software modules constituting it. This part is covered by sections 2 and 3.

Part II: Gives a detailed technical decription in terms of services, functions, protocols and interfaces to each of the software components. Sections 4 to 7 constitute part II.

The Data Transmission System is based on both the RC5000 MegaSwitch and the RC5000 Network Switch hardware platforms. As the software components of the Data Transmission System for both platforms are very similar, if not identical, no specific references are made to either platforms, except when important differences exist.

2. System Overview

The objective of the RcPAX Data Transmission System is to provide an efficient mechanism to transfer data quickly and reliably between users of the system. Users are connected to Access Points, through which data enter and leave the Data Transmission System, see fig. 1.

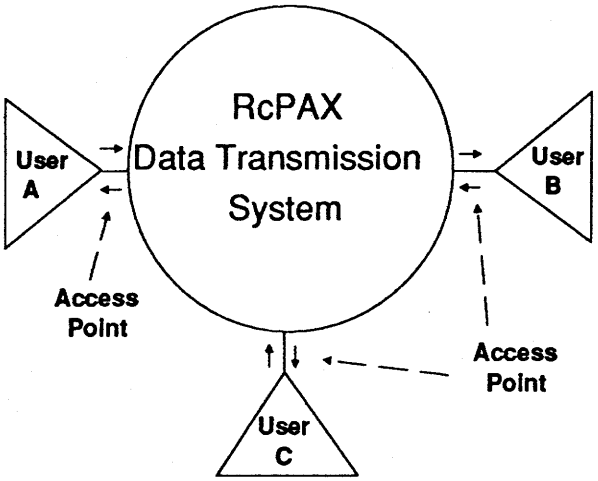


Figure 1 The Data Transmission System

To the Data Transmisison System two categories of users exist, external and internal, see fig. 2.

- External users are connected to the Data Transmission System via physical links. The communication procedures between an external user and the Data Transmission System are ruled by established standards such as the CCITT Recommendation X.25.

- Internal users are parts of the RcPAX System itself which requires means of exchanging data with other users. Internal users are attached to the Data Transmission System via an internal software interface defined specifically for this purpose.

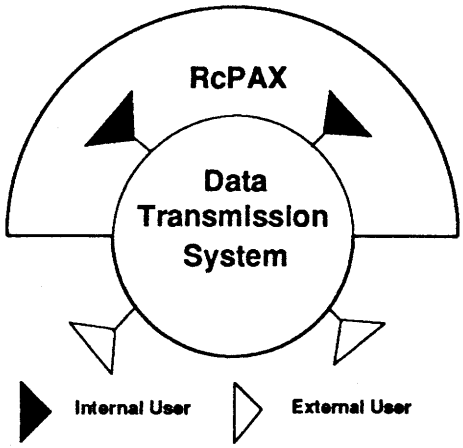


Figure 2 External/Internal Users

The Data Transmission System supports unrestricted communication between both internal and external users.

Physically, the Data Transmission System is made of a mesh network of communication computers interconnected by a number of transmission links. Both the network nodes (the communication computers) and the transmission links may be of different types and capacity. All together the nodes and links form the distributed physical platform for the Data Transmission System.

This physical platform built of hardware components is the basis for the software components residing in the network nodes. In conjunction with the hardware components the software components cooperate to form the total functionality of the Data Transmission System.

The communication procedures on all transmission links are controlled by a link level protocol, typically an HDLC protocol such as LAPB. Normally, the link level protocol (as well as the physical level) is handled by special communication controllers or hardware devices. Although much software may be involved to provide these vital services for the Data Transmission System, such software is not in this context considered as software components of the Data Transmission System.

In general, the software components are the parts of the Data Transmission System responsible for establishing the different communication services offered by the Data Transmission System. To this end, a very large number of different tasks have to be performed by the software components.

To simplify both the structure and the implementation of the Data Transmission System, these tasks have been grouped so that each group has a specific, well-defined functionality. This functionality is expressed as the services offered to other groups of tasks.

The overall structure of the Data Transmission System is that of the traditional layered type where new services are built on top of, and rely on other services.

This structuring principle has been adapted from ISO's Reference Model for Open Systems Interconnection, the OSI Reference Model.

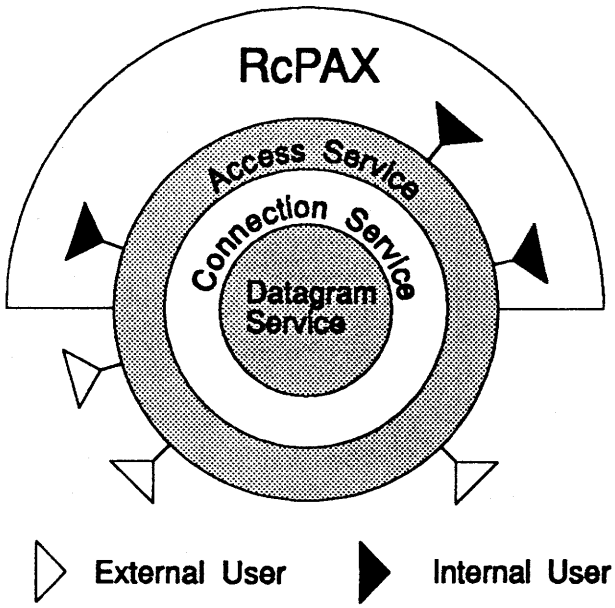


Figure 3 Services within the Data Transmission System

Three main services are identified within the structure of the Data Transmission System:

- datagram service
- connection service
- access service(s)

as shown fig. 3.

Additionally, the concept of "associated services" is identified, see below.

Datagram service
The kernel of the Data Transmission System is the packet switching network. Packets are sent from source to destination nodes based on address information present in each packet. The packet switching network handles the packets independently of each other. This means that packets sent between the same source/destination pair may take different paths due to changes in the network topology, see fig. 4. The network automatically adapts to changes in network topology while maintaining the very basic virtue of packet switching: sharing of the network resources.

The datagram service can be defined as the means to move datagrams, i.e. packets, between the network nodes. However, due to the nature of the packet switching network, there is no datagram delivery guarantee, i.e. datagrams may be lost. Also, datagrams may be duplicated and delivered out of sequence. In short, the packet switching network will do its best to deliver a datagram, but it does not guarantee proper delivery.

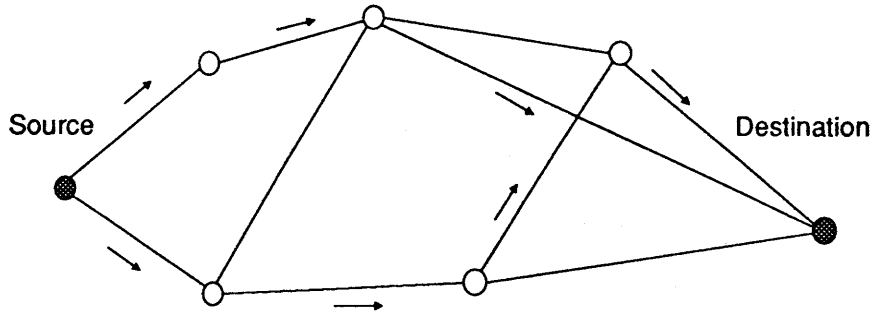


Figure 4 Datagram Service

Connection service
In most cases the datagram service by itself, due to the built-in deficiencies of the packet switching strategy, cannot meet the reliability requirements for data transfer between users of the Data Transmission System. To satisfy such needs, the connection service is used.

The connection service is built on top of the datagram service and provides a connection-oriented communication service on an end-to-end basis between the source and destination nodes, see fig.5. The end- to-end connection spans any number of hops in the packet switching network.

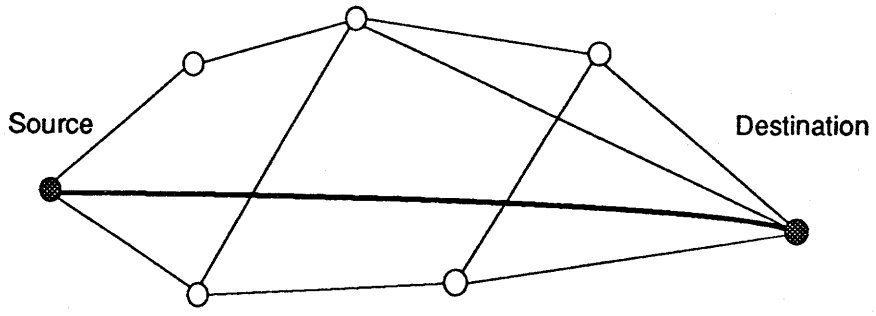


Figure 5 Connection Service

On such a connection, the connection service provides:

- *Delivery guarantee.* All data packets are delivered to their destination (or the source is informed by a connection reset).
- *Sequence guarantee.* Data packets are delivered to their destination in the same sequence as sent by the source.
- *Flow control mechanisms.* The connection service offers means to control the end-to-end flow of data packets.

To provide these services, the connection service must take care of:

- Packet retransmission in case the packet is dropped by the datagram service.
- Packet resequencing when packets arrive out of sequence at the destination.
- Packet disposal if packet duplication has taken place in the datagram service.

Access service
The access service is the part of the Data Transmission System which is oriented towards the users. Built on top of the connection service, the access service handles the communication procedures between the Data Transmission System and its users. As such, the access service can be seen to perform the adaptation of the "raw" data transfer service, provided by the connection service, to the users' needs, as expressed in the relevant standards, e.g. X.25.

The functionalities required by the users are much more than just raw data transfer: addressing functions, user groups, call barring etc. etc. Such functions are all taken care of by the access service.

The Data Transmission System supports a number of different access services to the users:

- X.25 access service for directly connected packet mode DTEs according to the CCITT Recommendation X.25.
- X.32 access service for packet mode DTEs connected via dial up lines according to the CCITT Recommendation X.32 (and X.25).
- X.75 access service for directly connected public network according to the CCITT Recommendation X.75.
- Internal access service for software components within the RcPAX System itself, e.g. the PAD X.28 start/stop Terminal Service.

The different types of access service are illustrated fig. 6.

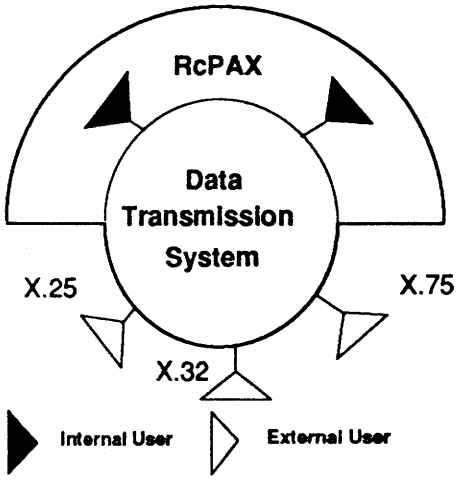


Figure 6 Access Services

Associated services
As the Data Transmission System is an integrated part of the RcPAX system, the datagram, connection and access services are not self-contained in the sense that more functionality is needed to complete an operating Data Transmission System. Such services are called associated services in this context.

Management service
The Data Transmission System is manageable and as thus accessible to the Network Management System, see ref. 2, enabling a Network Operator to perform a number of surveillance and control operations upon the Data Transmission System. The mechanisms making the Data Transmission System manageable are common for all RcPAX software components residing in network nodes.

Directory information service
The access service part of the Data Transmission System requires some directory information services to perform a number of functions such as verification of Network User Identification, inter-network routing to public networks (X.75) etc. This service provides the means to retrieve information associated with an identifiable data object already stored in the directory database. The information retrieved by the access service is put into the directory database by the network operator. The directory database is implemented in a distributed fashion with parts locatable anywhere in the network.

Address translation service
In order to provide logical addressing facilities and inter-network routing to private networks (X.25), the access service of the Data Transmission System requires means of mapping between addressing spaces. This is offered by the address translation service. The address translation service is by itself based on the directory information service.

Buffer management service
The access part of the Data Transmission System requires appropriate means for buffer management to ensure efficient and flexible use of buffer resources. Such means are provided by the buffer management service supporting both restricted and shared use of the buffer resources.

3. Software Components

The datagram, connection and access services sketched in the preceding section are implemented by a number of software components. In the aggregate these software components constitute the software part of the Data Transmission System.

Listed according to the related service the software components are:

Datagram service: ROUTER

Connection service: LIC (Logical Internal Channel)

Access services:
- DCE
- STE
- IDTE (Internal DTE)

Each software component is introduced in the following sections, while detailed descriptions are given in part II of this document.

The software components constituting the associated services of the Data Transmission System are also introduced:

Associated service: NCP (Network Control Probe)
- DIS (Directory Information Server)
- ATS (Address Translation Server)
- SL (Session Layer)
- TL (Transport Layer)
- POOLMAN (Pool Manager)

As these software components described in detail elsewhere (see refs. 4, 5) they are not included in part II of this document.

In the rest of this document the terms software component, activation and entity are used in a specific way implying a relationship between them.

A software component can in the "real world" be considered as consisting at some pieces of object code having some inbuild potential functionality; much like a hardware component before it is powered up.

When a software component is activated an entity of that software component is created. An entity can be seen as a "running" software component, like a powered up hardware component. Softwarewise, an entity can consist of a number of software processes possibly running on a number of hardware components, all depending on the design of the software component in question.

Unlike a hardware component, a software component may be "powered up", i.e. activated, more than once. This means that more than one entity of a software component can exist simultaneously in the same node.

3. 1. Datagram Service

The datagram service is implemented by the ROUTER software component. A ROUTER entity is present in every network node. The ROUTER entity is responsible for the switching of packets to/from the neighbour nodes and to/from the node itself. As such all the ROUTER entities in all the network nodes constitute the packet switching system.

The structure of a ROUTER entity can be illustrated as a set of "connectors" through which packets arrive/leave the ROUTER entity, and a kernel switching packets between the connectors, see fig. 7.

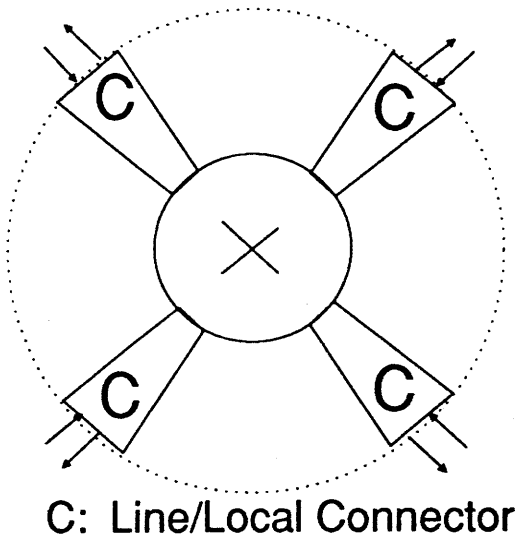


Figure 7 ROUTER Structure

Two types of connectors are distinguished:

- A local connector allows packets to/from the node itself to leave/enter the packet switching network, i.e. a local connector represents an entry point to the datagram service within the node itself.

- A line connector allows packets to be exchanged between neighbouring nodes of the packet switching network. The line connectors are typically based on a Link level protocol, e.g. an HDLC protocol, to communicate with a neighbour node, see fig. 8.

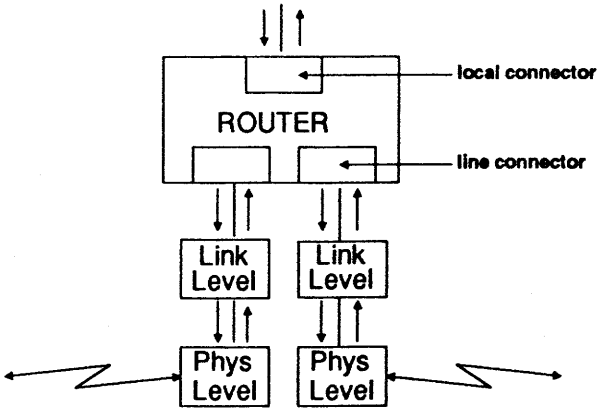


Figure 8 ROUTER and Link Levels

Among others, the ROUTER entities in neighbour nodes exchange routing information using the Neighbour Node Protocol (NNP) (see fig. 9). Using the NNP the ROUTER entities in the network nodes cooperate to handle the RcPAX routing algorithm. This decentralized algorithm ensures that packets normally are sent along the shortest path to their destination node, and, at the same time, the algorithm adapts dynamically to changes in network topology without intervention from the Network Operator.

In addition to its very basic function in the Data Transmission System, the ROUTER also has a role in the RcPAX Load System in relation to neighbour nodes requests for software load over the network, see ref. 3.

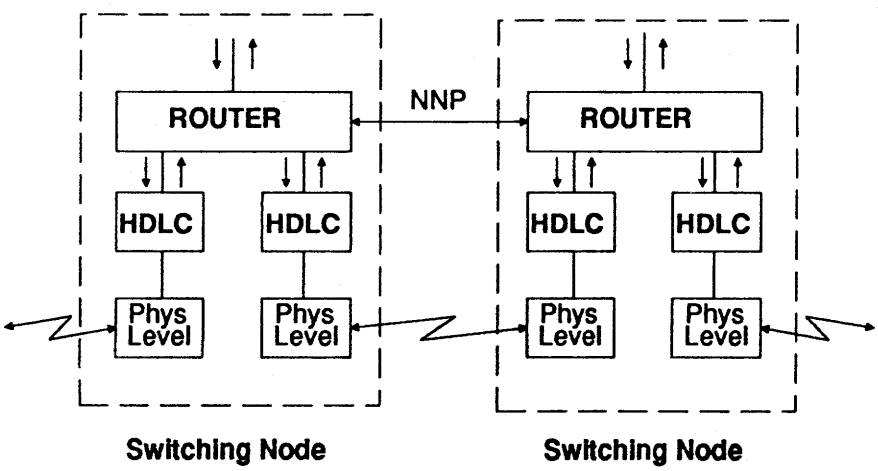


Figure 9 Neighbour Node Protocol

3. 2. Connection Service

The LIC software component implements the connection service. An activation of the LIC software component, a LIC entity, is present for each Access Point to the Data Transmission System, i.e. a number of LIC entities may exist within a node.

A LIC entity makes use of the datagram service as provided by the ROUTER entity, see fig. 10.

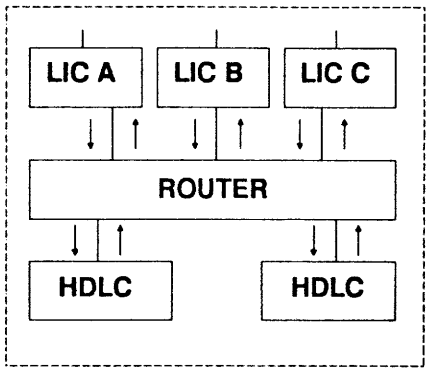


Figure 10 LICs within a node

To determine to which LIC entity incoming packets shall be delivered, the ROUTER entity uses the so-called extension part of the address information present in the packet, i.e. each LIC entity is unambiguously addressable.

The LIC entities provide the connection service on so-called logical internal channels established between them. Each LIC entity can handle many logical internal channels to a lot of LIC entities, see fig. 11.

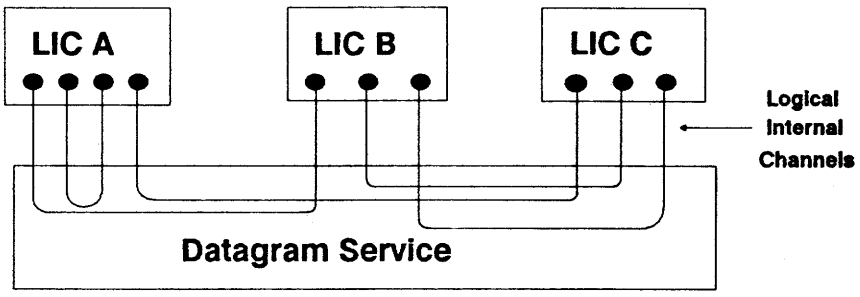


Figure 11 Logical Internal Channels

The LIC entities on each end of a logical internal channel cooperate to provide the connection service: delivery and sequence guarantee and flow control. To this end they run the internal LIC protocol between them, see fig. 12

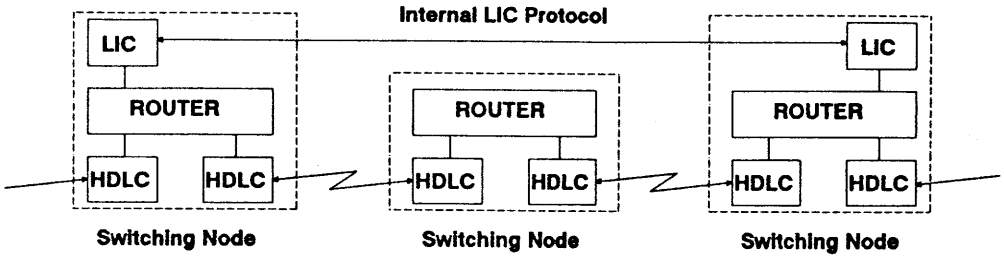


Figure 12 Internal LIC Protocol

As shown fig. 12 the internal LIC protocol operates on an end-to-end basis spanning the whole packet switching network.

The internal LIC protocol contains elements to control the internal logical channels (establishment, clearing etc.). To provide the delivery and sequencing guarantee the internal LIC protocol uses a packet sequence numbering scheme combined with packet acknowledgments. The flow control is based on a simple window/credit mechanism.

3. 3. Access Services

Three access service software components are used to implement the four access services described in section 2:

- the DCE software component implements both the X.25 and the X.32 access services
- the X.75 access services are handled by the STE software component
- the IDTE software component takes care of the internal access service

New access software components may come in the future, as new mechanisms for access to the Data Transmission System may be developed.

For each Access Point of the Data Transmission System there is an activation of one of the three software components, i.e. a DCE, STE or IDTE entity. Each entity runs on top of a corresponding LIC entity making use of the connection

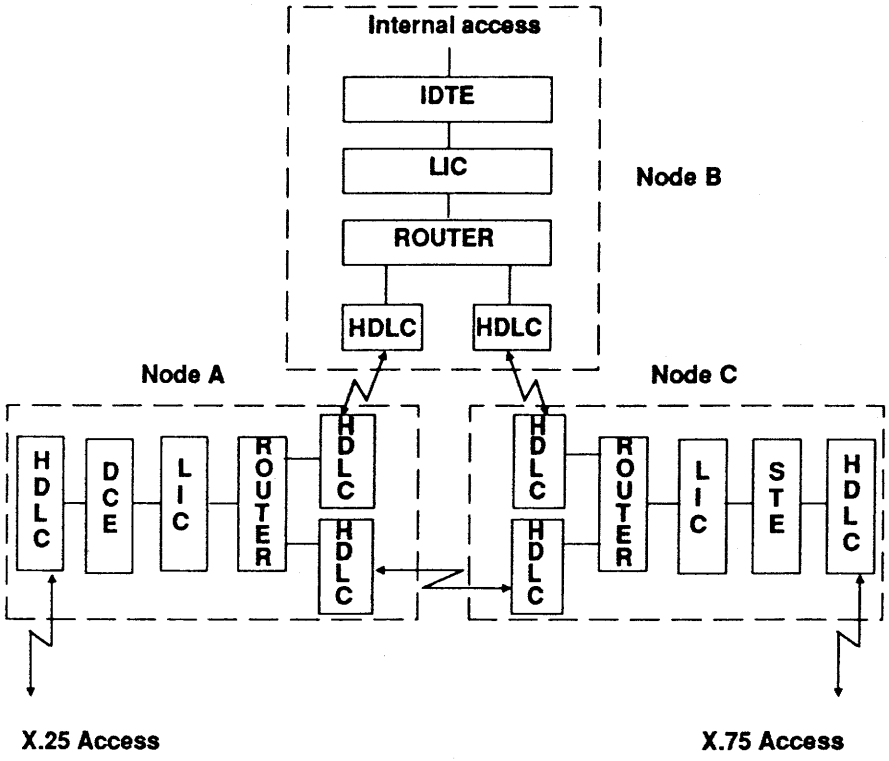


Figure 13 Access Modules

A node will normally contain a number of access service entities of different types.

For the provided access services so-called Virtual Circuits are used to carry data between users of the Data Transmission System. Each user may use many Virtual Circuits simultaneously. The access entities map each Virtual Circuit on exactly one logical internal channel as provided by the connection service (LIC entity). This ensures data integrity on the Virtual Circuits, even in case of problems within the packet switching network.

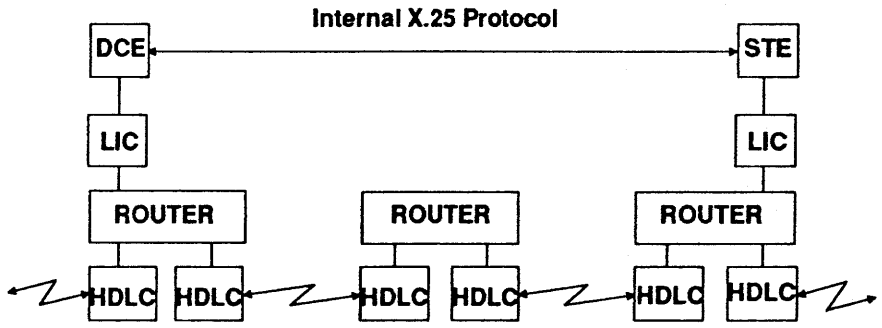


Figure 14 The Internal X.25 Protocol

The Data Transmission System supports the establishment of Virtual Circuits between any pair of users disregarding their type of Access Point, i.e. DCE, STE or IDTE. To this end, and to handle a large number of functions additional to data transfer between users, the access modules interwork using an internal X.25 protocol, see fig. 14.

The internal X.25 protocol resembles very much the X.25 level 3 protocol, but is in principle proprietary to the Data Transmission System and is subject to additions as new access service requirements may arise.

3. 4. Associated Services

A number of software components are used to implement the associated services as outlined in section 2. Some of these software components can rightfully be considered as having a much wider scope with the RcPAX System than just the Data Transmission System. However, as the Data Transmission System as a whole relies on these services, their roles in the Data Transmission System are outlined in the following.

The management and directory information services are implemented in the RcPAX system as distributed applications. This implies that the entities spread throughout the RcPAX System (both Network Management Centres and the network nodes themselves) require some common communication service to exchange specific information between them. The session communication services serve this purpose.

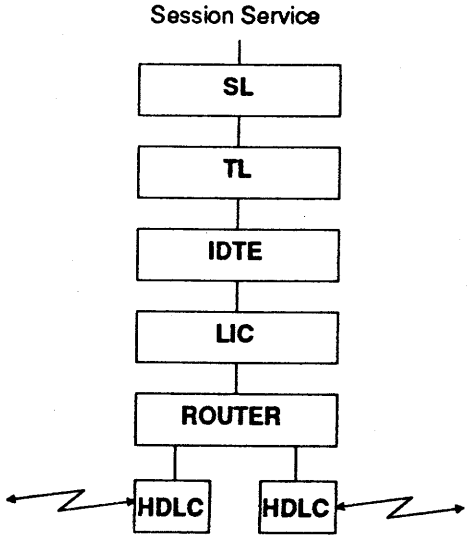


Figure 15 Session Service

The session service in the network nodes is implemented by the SL software component based on the TL software component. The TL software component relies in turn of the internal access service provided by the Data Transmission System itself, see fig. 15.

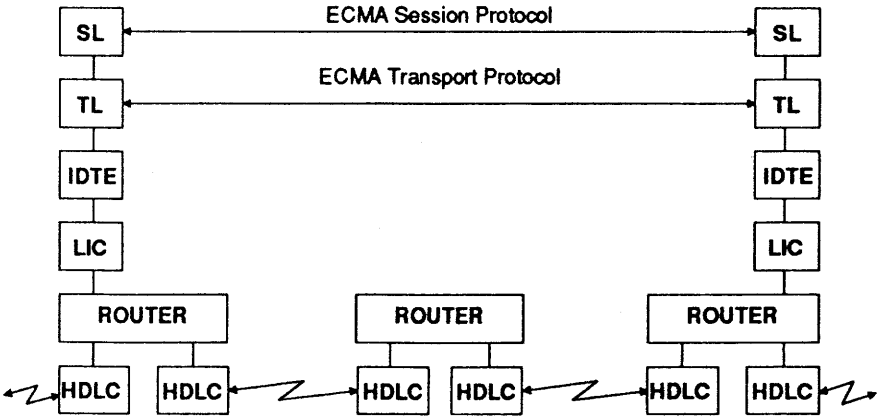


Figure 16 The TL/SL Protocol

To one activation of the SL software component, i.e. an SL entity, corresponds exactly one activation of the TL software component, i.e. a TL entity.

The communication protocols used between the TL/SL entities are the ECMA Transport and Session Layer Protocols respectively, see fig. 16.

It should be mentioned that the SL software component makes use of the directory information service to support a logical addressing scheme.

Management service
The NCP software component is a central part of the RcPAX Network Management System (NMS). An activation of the NCP software component, the NCP entity, is mandatory in all network nodes. The NCP entity is common to all parts of the RcPAX System represented in that node.

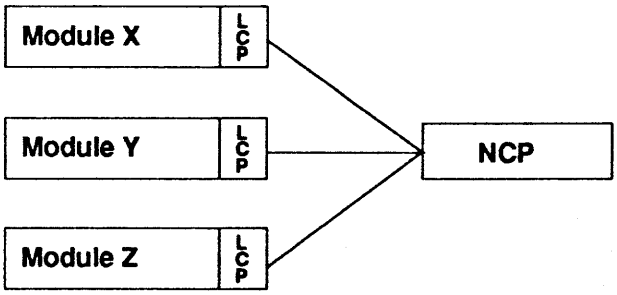


Figure 17 The NCP/LCP Interworking

The NCP entity may be seen as the coupling between the manageable entities within the node and the other parts of the NMS. Each manageable entity contains an LCP (Local Control Probe) for this purpose, see fig. 17.

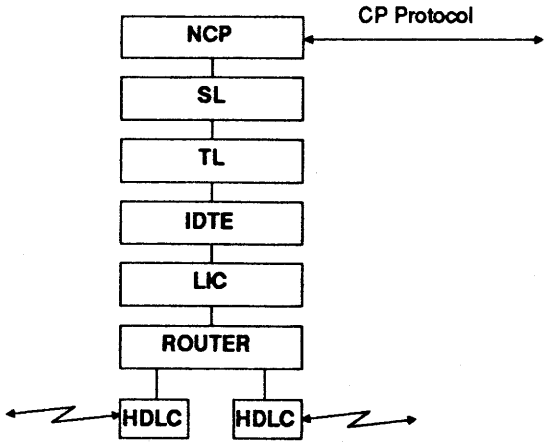


Figure 18 The CP Protocol

The interworking between the NCP entity and the LCPs within a node is provided by local means.

The NCP entity communicates with the other parts of the NMS using the CP (Center Protocol) protocol on top of the session service, see fig. 18. See ref. 2 for a description of other parts of the NMS and the role of the CP protocol).

Directory information service
This associated service is provided by the DIS software component represented as a single activation, the DIS entity, within each node holding entities requiring access to the directory information service.

The access to the DIS entity within a node is provided by local means, e.g. from the access entities DCE, X75, IDTE.

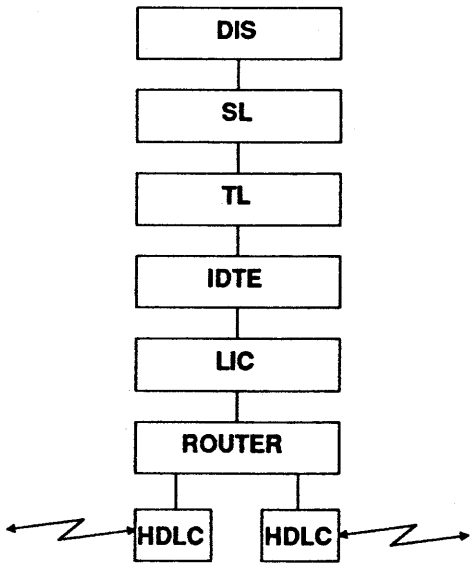


Figure 19 The DIS Entity

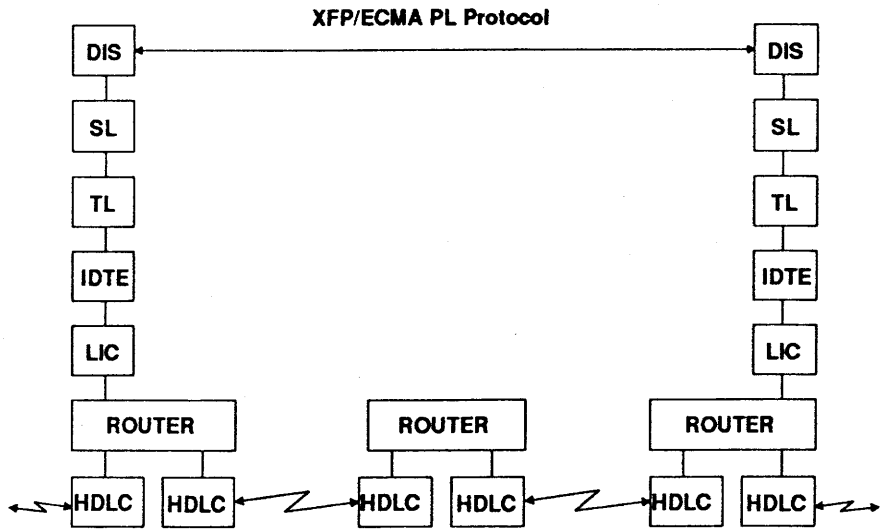


Figure 20 The XFP/ECMA PL Protocols

The DIS entity runs on top of the SL module making use of the session service, see fig. 19.

To supply the directory information service, the DIS entity within a node has to interwork with DIS entities within other nodes, e.g. exchange enquiries of named data object. To transfer such enquiries (and the corresponding answers) the DIS software component implements the XFP Protocol and the ECMA Generic Presentation Protocol, see fig. 20.

The data objects and the associated information retrievable through the directory information service may be placed in any DIS entity in the network according to a naming tree making the data objects globally identifiable. By configuration conventions, two types of DIS entities are normally referred to, Master-DIS entities and Slave-DIS entities. The Master-DIS entities hold the data objects while the Slave-DIS entities hold no data objects, and serves then 'only' as access points to the directory information system. It should be noted that the DIS software component is used both as Master- and Slave-DIS entities; it is only its use that differs.

Address translation service
The ATS software component implements the address translation service. An activation of the ATS software component, an ATS entity, is present in each node when the address translation service is used, i.e. in nodes with one or more DCE, STE or IDTE entities supporting some address translation facility.

The interworking between the ATS entity and the address translation service users within a node is provided by local means.

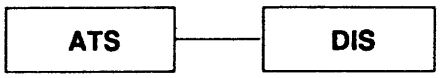


Figure 21 The ATS/DIS Entities

The address translation service is built on top of the directory information service, which implies that the ATS module requires access to the DIS entity, see fig. 21.

Buffer management service
The buffer management service is implemented by the POOLMAN software component. The DCE and STE access service entities make use of this service. Therefore, one or more activations of the POOLMAN software components, POOLMAN entities, exist in nodes with DCE and/or STE entities.

The access to a POOLMAN entity within a node from a DCE/STE entity is provided by local means, see fig. 22

Figure 22 The POOLMAN Entity

Each DCE/STE entity makes use of one POOLMAN entity. However, more DCEs and/or STEs may access the same POOLMAN entity allowing buffers to be shared between them. The POOLMAN software component implementes this buffersharing in conjunction with mechanisms providing DCE/STE entities to reserve a number of buffers for private use. One POOLMAN entity handles buffers of the same size.

The associated services are provided by entities of the different software components as described in the preceeding sections. Fig. 23 illustrates the usage of all those services by the access service of the Data Transmission System.

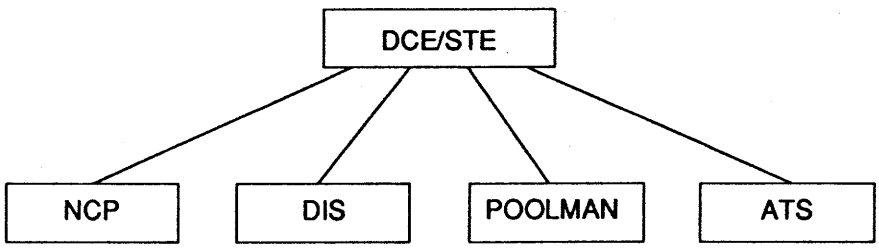


Figure 23 Associated Services Entities

4. ROUTER

The datagram service is implemented by the ROUTER software component. The ROUTER software component is implemented in two different versions. A version, mainly programmed in occam, is running in the RC5000 Mega-Switch node. Another version is programmed in Real Time Pascal (RTP) and is running in the RC5000 Network Switch node. Even though the two versions cooperate to implement the same routing algorithm, differences do exist.

The ROUTER software is implemented by several communicating processes: one Router, a LineConnector for each line and some LocalConnectors. Router is responsible for the routing algorithm, LineConnector controls the line and LocalConnector provides the access points seen by users of the datagram service.

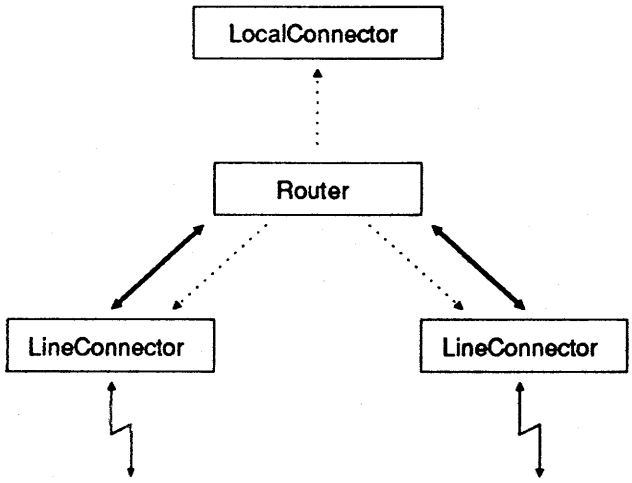


Figure 24 Flow of Routing Information

Router maintains and distributes routing information. Distance vectors are exchanged with the neighbour Routers and distance vectors are sent to the connectors, in order to make these able to route received datagrams to the next Connector. Router also exchange routing information with the neighbour Router, see fig. 24.

The actual routing of datagrams is performed by the LineConnectors and LocalConnectors, see fig. 25.

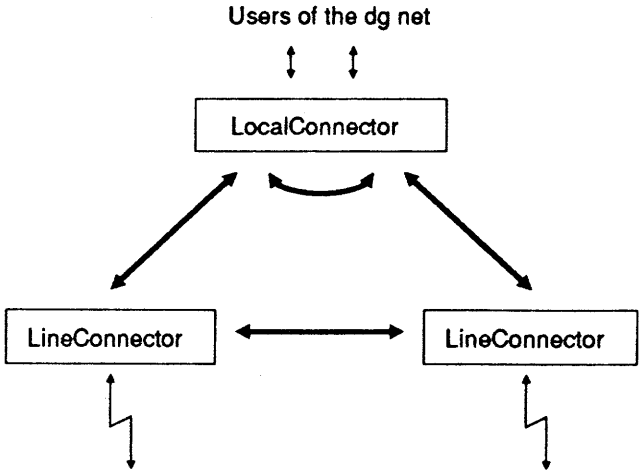


Figure 25 Flow of Datagrams

4. 1. Routing Algorithm

The RcPAX datagram network is implemented by a distributed, hierarchical, adaptive routing algorithm. The datagram network includes lines and nodes with different capacities.

The concept of a datagram network means that datagrams are routed individually. Datagrams will be routed along an alternative path if a line or node on the shortest path fails; no higher level connections are broken as long as alternative paths exist.

A datagram network does not guarantee the delivery of all datagrams and do not necessarily preserve the sequence of datagrams delivered. To overcome from these problems, and to implement a flow control reducing the risk of bursts in the network, an internal virtual circuit protocol, the Logical Internal Channel protocol (LIC), is used. This protocol uses the datagram network and deals with internal connection establishment and removal, and with retransmission from the sender and resequencing at the receiver of datagrams.

The routing function is distributed and adapts to changes in the network topology automatically. The Routers report current routes to each neighbour Router. On discovering a topological change or receiving routing information from a neighbour, new routes are calculated, and changes in routes are distributed to the neighbours.

Addressing in the RcPAX data transmission system is hierarchical. Each node in RcPAX has been assigned a 3-byte address. This address is divided in 3 hierarchies, net, region and node, with a one-byte address each. The net level is the highest level of addressing followed by the region level and the node level. Internally in a node a fourth addressing hierarchy, the extension address, is used. Users of the datagram service is identified by a datagram address, consisting of the net, region and node address of the node where the user is connected, followed by a two-byte extension address, which is local to the node.

RcPAX uses hierarchical routing. This reduces the amount of routing information stored in a node and reduces the amount of routing information exchanged due to a topological change. RcPAX employs four levels of routing which are following the addressing hierarchy: net, region, node and extension. However, in the RC5000 MegaSwitch nodes, the region and node level have been joined, in order to make it possible to configure a network more freely.

A shortest-path routing is used in carrying out the routing function. To each line and node a cost, or the relative desirability of passing the line/node, is assigned. This cost is called the line resistance and the node resistance respectively. The cost of a path in the network is the sum of the line resistances and node resistances along the path (excluding the node resistance of the source and destination node). The distance between two nodes is the minimum of the cost of all paths between the nodes.

In determining the path to follow for a datagram, a path with the least cost is chosen. To balance the load, traffic is spread on several paths, when more than one path with the least cost to the destination node exists.

4. 2. Interface

The datagram service is provided at a LocalConnector. Within a node, users of the datagram service are distinguished by the extension address.

Each user of the datagram network connects to the datagram network with a 2-byte extension address. The extension address must be unique within the node. Several users may be connected at one LocalConnector.

The datagram network can forward datagrams between any two pairs of extensions, i.e. deliver datagrams between any pair of modules connected to the datagram network.

The primitives described in the following are the ones used by the users when accessing the datagram interface.

Connect Extension
This operation is used to connect a user with a certain extension address to the datagram net.

Disconnect Extension
A certain extension is disconnected from the datagram net.

Transmit Packet
A datagram is sent into the network. The operation contains the destination datagram address and the origination datagram address, as well as the user data to be transmitted.

Receive Packet
This operation is queued at the LocalConnector, until a datagram for this extension is received from the network. When returned it contains the received datagram.

Reset
All queued Receive Packet operations from this extension are returned.

Read Own Address
Returns the net, region and node address of this node.

The datagram network supports transmission of datagrams of any size through the network.

Internally in the datagram network there is a fixed upper limit on the size of datagrams that can be forwarded.

To overcome from this the LocalConnector process supports fragmentation of datagrams entered at an extension, and assembling of the datagrams at the destination end before delivered at the destination extension.

The fragmentation and corresponding assembling is transparent to the user of the datagram service.

4. 3. Protocols

The Router protocols use the services from a link level protocol. The link level protocol used is the CCITT X.25/2 LAPB protocol, extended with an HDLC XID-subprotocol in order to negotiate the HDLC address and sequence numbering, see fig. 26.

A small protocol called Connector Data Protocol (CDP) is used for switching datagrams between the LineConnectors, see fig. 26.

The datagrams to be switched contain a router header. This router header contains among other things the destination datagram address, the remaining expected distance (see section 4.4.3), a type for statistic purpose (see section 4.5.3), the length of the data and an end-to-end CRC (usually not computed, see section 4.5.2).

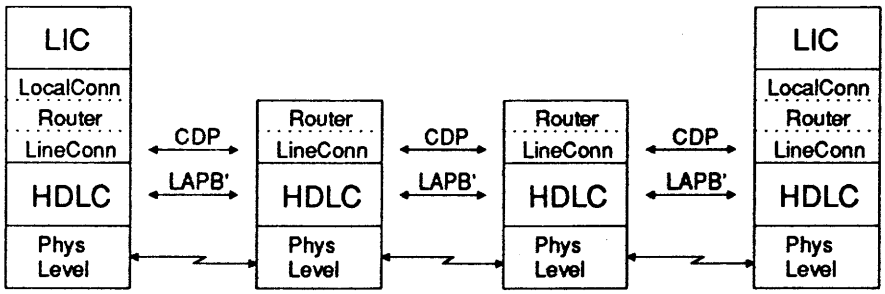


Figure 26 Connector Data Protocol

In RcPAX a Neighbour Node Protocol (NNP) is used for exchanging information between neighbour Router processes, see fig. 27.

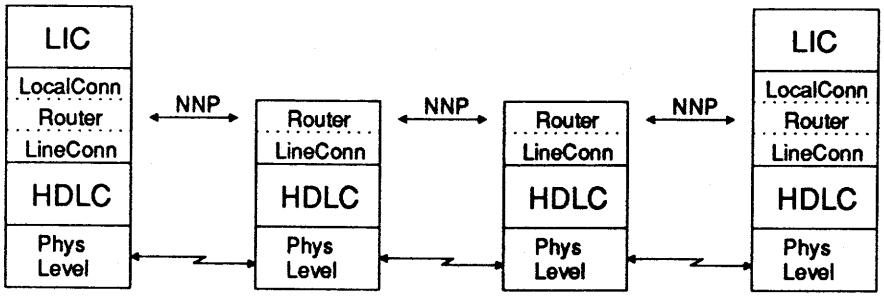


Figure 27 Neighbour Node Protocol

The Neighbour Node Protocol (NNP) contains the packet types: Load Request, Line Information, Test, Range Vector, Network Clock and Broadcast. The parameters in an NNP packet are type/length/value (TLV) coded, allowing different parameters to occur in the same packet and flexible for extensions.

Load Request is used for generating a request for autoload of the node sending the load request.

Line Information is used for negotiating the version of the NNP to be used and exchanging information about the nodes and the line, such as node address, class of node (RC5000 MegaSwitch or RC5000 Network Switch) and line identification. Additionally, Line Information is used for implementing a subprotocol of the NNP, called the handshake protocol. The handshake protocol is used to check the quality of the line and estimating and negotiating the cost of the line (line resistance) before the line is used as a router line. Furthermore, the handshake protocol is used to prevent lines from going up and down repeatedly.

Test packets may be used during the handshake protocol to estimate the cost of the line.

Range Vector is used for exchanging distance vectors used for maintaining the routing tables. Only changes in the distance are exchanged.

Network Clock is used for synchronizing a common network clock.

Broadcast is used for implementing a broadcast algorithm.

To implement the fragmentation and corresponding assembling of datagrams a fragment protocol between the LocalConnectors is used, see fig. 28.

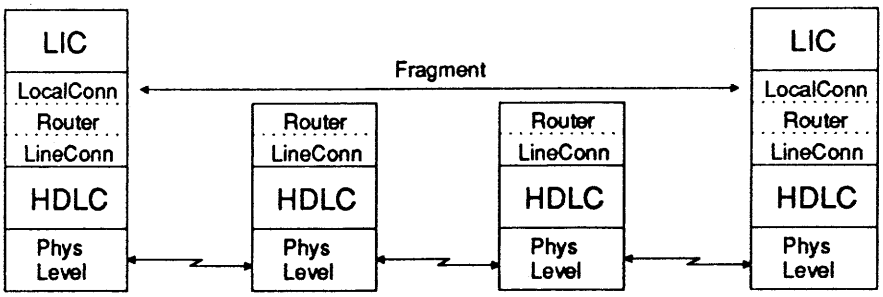


Figure 28 Fragment Protocol

4. 4. Routing Service

Details on the main service provided by the datagram network, the routing of packets, are given in this section.

4. 4. 1. Routing Hierarchy

The routing in RcPAX is hierarchical. This imposes some restrictions on which addresses to assign to nodes, considering the configuration of the network.

In RC5000 MegaSwitch nodes the routing is hierarchical following the hierarchy net, node and extension. The region and node address is used as one (node) level in RC5000 MegaSwitch nodes.

In RC5000 nodes the routing is hierarchical following the addressing hierarchy net, region, node and extension.

In RcPAX the following restrictions on the addresses/configuration must be met, in order to guarantee the delivery of packets.

- At the net level the network must be connected; that is, there must be a path between any two nodes with the same net address, without going through nodes with another net address, see fig. 29.

- The regions need not be connected, but between any two nodes with the same net and region address there must be a path either not leaving the net and region address, or when leaving the region address, only passes through RC5000 MegaSwitch nodes, see fig. 30.

The same routing function is used at the net, region and node levels, while extension level routing is local to a node.

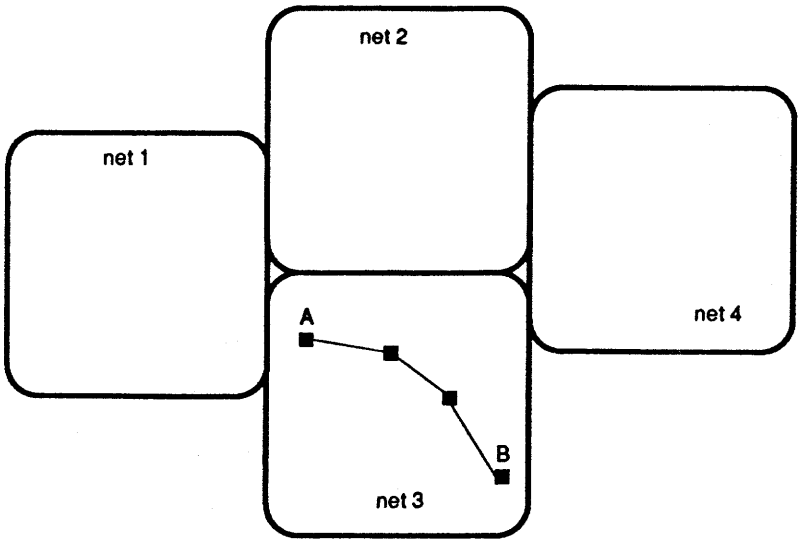


Figure 29 Net must be connected

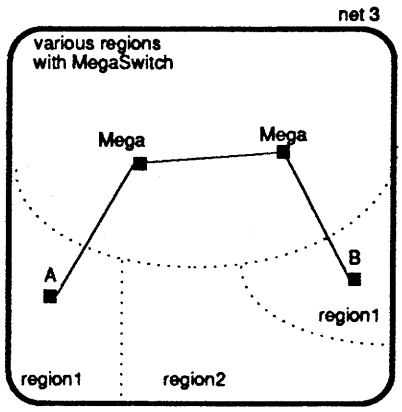


Figure 30 Region connected through MegaSwitch

Datagrams are routed in the ordering net, region, node and extension. Some levels may be skipped, but routing is never returned to a higher level.

Routing in a network consisting both of RC5000 MegaSwitch and of RC5000 Network Switch nodes is done in the following way, which is illustrated in fig. 31.

A datagram is first routed along a shortest path to the destination net address, i.e. routed towards the nearest node with the destination net address. From the destination net it is examined whether a path to the destination node is known at the node level. If this is so, the path is followed to the destination. If not, the packet is routed along a shortest path towards the destination region address, until the destination becomes known at the node level. This will happen no later than when the destination region is reached. From this point the datagram is routed at the node level and finally, when the destination node is reached, the datagram is routed to the destination extension.

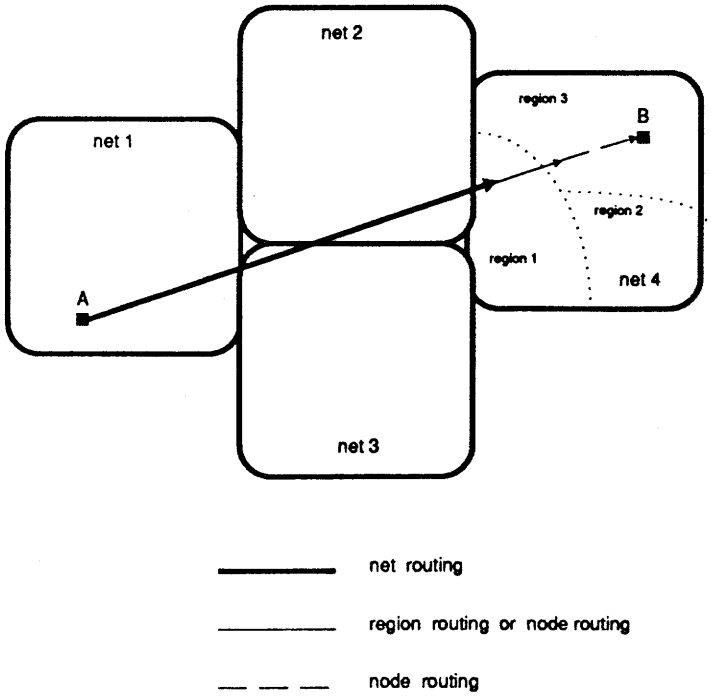


Figure 31 Routing Packets

In order to route datagrams, the following distance vectors are exchanged between two neighbour nodes (see fig. 32).

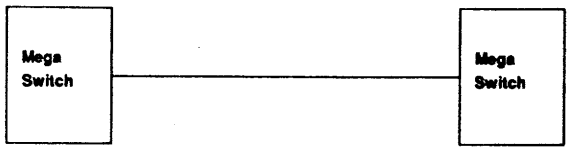
Between any two neighbour nodes, information about the net level is exchanged. Between any two nodes in the same net, information about the region level is exchanged. Between two RC5000 MegaSwitch nodes in the same net, information about all nodes in the net is exchanged at the node level. Between an RC5000 MegaSwitch and an RC5000 Network Switch node in the same net, information about all nodes in the RC5000 Network Switch nodes region is exchanged. Between two RC5000 Network Switch nodes in the same region, information about all nodes in the region is exchanged.

In this way RC5000 MegaSwitch nodes are informed on the node level about distances to all nodes in the regions which are connected to an RC5000 MegaSwitch node by a line. Distances to nodes in regions not connected to an RC5000 MegaSwitch by a line are only informed to RC5000 MegaSwitch nodes on the region level.

4. 4. 2. Routing Tables

The information for routing are stored in several tables. The routing tables contain the necessary information about the topology of the network. From the routing tables, distance and path vectors are calculated, these contain the information necessary to route a packet.

For each level of routing, a routing table is maintained.



net level
region level, if same net address
node level, if same net address

net level
region level, if same net address
node level for Network Switch's region (if same net address)

net level
region level, if same net address
node level, if same net and region address

Figure 32 Distance Vectors exchanged

The routing table for the net level is a two-dimensional matrix, with rows corresponding to net addresses and columns corresponding to lines at the node. Each entry (i,j) contains the shortest distance from this node to the net i (i.e. a node in net i) if the line j is used as first edge in the path, see fig. 33.

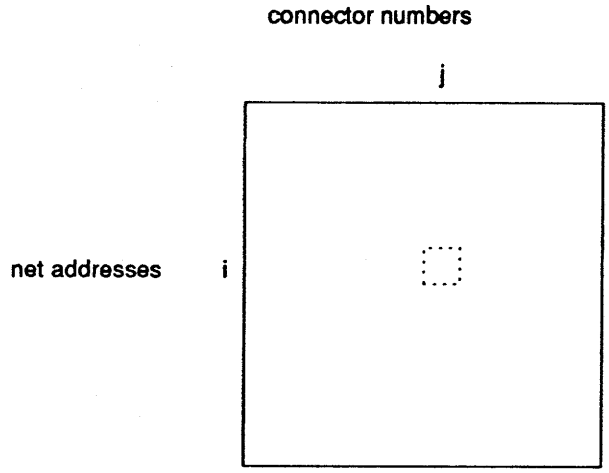


Figure 33 Routing Table for the Net Level

The routing table for the node level in RC5000 MegaSwitch is a two-dimensional matrix, with rows corresponding to nodes in this net and columns corresponding to lines at the node. Each entry (i,j), contains the distance from this node to the node i, if the line j is used as the first edge in the path.

For nodes belonging to regions not sharing a line to an RC5000 MegaSwitch node, the node level routing table is not updated, but will contain an infinite distance. Routing information for these nodes is stored in the region level routing table, which is maintained in RC5000 MegaSwitch nodes, too.

The region level routing table is a two-dimensional matrix, with rows corresponding to regions in this net and columns corresponding to lines at the node. Each entry (i,j) contains the shortest distance from this node to the region i (i.e. a node in region i), if the line j is used as first edge in the path.

The table for routing on the extension level is local to the node, and is not causing updating information to be exchanged with the neighbours. For each extension connected to the network by this node a table entry is maintained in the extension level routing table.

From the routing tables distance vectors are calculated for the net, region and node level whenever topological changes occur. The distance vectors contain the topology information sent to neighbours and are used when routing a datagram. Distance vectors are one-dimensional arrays, with the same rows i as the routing tables, and containing the shortest distance to the address i seen from this node (fig. 34).

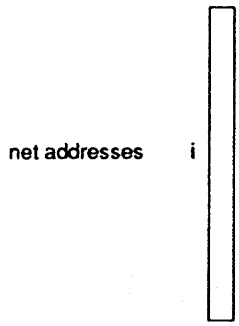


Figure 34 Distance Vectors for the Net Level at a Line

The distance vectors are not just calculated as the minimum of the row in the routing tables, but are modified in order to prevent routing a datagram back to the node it is received from. The modification takes into acount not just single lines between neighbours, but also parallel lines between neigbours. The modification significantly reduces the topology information exchanged when lines fail. Distance vectors are, because of the modifications, line specific, i.e. a different distance vector is calculated for each line in the node. The modified distance vectors are stored decentralized at each process controlling a line.

The amount of updating information exchanged due to a node failure or a line failure isolating nodes depends on the limit for considering the distance to a node infinitely. The iterative process of exchanging updates will continue until this limit is reached. To reduce the updating, an individual limit for infinite distance may be configured in every node for each of the levels, net, region and node.

When calculating distance vectors, corresponding path vectors are calculated, too. Path vectors are used when routing a datagram. Path vectors are two-dimensional matrices with the same rows as in the distance vectors and with 4 columns. Path vectors specify for each address 4 lines which all may be used as the first edge in a shortest path to the destination, i.e. paths with the distance specified in the distance vector. If less than 4 paths with the shortest distance are found, then the same line is specified more than once in a row in the path vectors. The path vectors are, as they closely follow the distance vectors, line specific, and are stored decentralized at each process controlling a line (fig. 35).

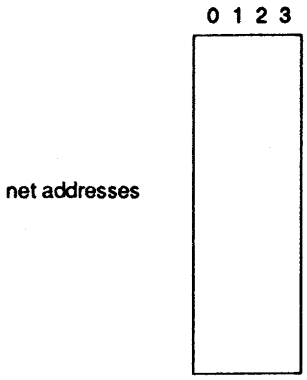


Figure 35 Path Vectors for the Net Level at a Line

4. 4. 3. Congestion Control

Several mechanisms are used to avoid congestion in the network.

First of all, the LIC protocol implements flow control. This flow control ensures that a single logical internal channel only has a limited amount of datagrams outstanding in the datagram network, at a time.

When routing tables are being updated, they may for a period be inconsistent, leading to loops in the routing. To avoid datagrams being routed into these loops and thereby causing congestion in this part of the network, every datagram is marked with the remaining expected distance to the destination. When a datagram is routed, the actual shortest distance to the destination is compared with the expected distance from the router header. The datagram is dropped if the distance is increasing.

Bursts may occur in any network. To avoid bursts causing the throughput of the network to be reduced, a very simple mechanism is used, as the datagrams are simply dropped.

For the RC5000 MegaSwitch the receive and transmit storage at each line is finite and is exclusively used for that line. When datagrams are to be transmitted, they are dropped if the transmit storage at this moment is filled. When datagrams are received from the line, they are immediately forwarded to the transmit area of the next Connector. This means that the receiver will not by backward pressure reduce the transmission capacity of the line. Hereby congestion is tried to be kept locally, not spread around by rerouting datagrams or reducing line capacity.

As a line failure will cause an increase in the exchange of topological updates which may not be ignored, it is for a very short period tried to reestablish the line, before the line failure is informed throughout the network. If the line is reestablished successfully, topological updates only have to be exchanged over this line. Mechanisms are used, to ensure that a line is not repeatedly failing without notification.

In order never to overload the network with topological updates, a flow control is implemented. For each line a flow control timer may be configured. This specifies a forced timeout between sending packets with topological updates over the line.

If the network is permanently overloaded, the nodes may individually be configured to allow less data entering the network. For each node a datagram flow control is implemented, constraining the access traffic in the node to an individual maximum of datagrams per second. This facility is not used automatically, but may be used by the network operator when overloads occur, or may be used initially when some access nodes are configured.

4. 4. 4. Estimating Costs

The cost of passing a node, the node resistance, is set to a value individually for each node. For RC5000 MegaSwitch nodes it may be configured by the NMC, while it for RC5000 Network Switch nodes is set to zero.

The cost of passing a line, the line resistance, is estimated separately by the processes at each end of the line. During the handshake protocol a common line resistance is negotiated from the estimated values.

The line resistance may be estimated in three different ways. The method for estimating the line resistance is configured from the NMC.

Firstly, the line resistance may be set from the NMC.

Secondly, the line resistance may be a function of the line speed. The function may be configured from the NMC, but a default function is provided. The line speed is automatically determined by the LineConnector.

Thirdly, the line resistance may be a function of an actually measured delay over the line. During the handshake protocol, NNP Test packets may be sent over the line and thereby measuring the time delay on the line. Parameters for measuring the delay and the function used to convert the measured value to a line resistance, may be configured from the NMC.

It has shown to be useful to estimate line resistance as a function of line speed. In networks where lines with a short and a long delay are mixed, e.g. networks with terrestrial and satellite lines, it may, however, be advisable for some lines to estimate line resistance from an actually measured delay over the line.

4. 4. 5. Resource Allocation

The buffer resources are handled differently in the two nodes RC5000 MegaSwitch and RC5000 Network Switch. The differences are mainly due to the different hardware and software environment.

In the LocalConnector, both in RC5000 MegaSwitch and RC5000 Network Switch, users of the datagram network provide receive buffers, which are queued until a datagram arrives from the net. When a datagram has arrived from the network, the datagram is copied into the user's receive buffer, which is then returned to the user. If no receive buffer from the user is queued, or received within a short time, then the datagram arrived from the network is dropped.

In RC5000 MegaSwitch the LineConnectors are run on different boards, without any shared memory.

Each LineConenctor is configured with a receive and a transmit area. Datagrams are stored in the receive area only a short while, they are forwarded to the next Connector as soon as they have been completely received. Datagrams are stored in the transmit area until they have been transmitted and an acknowledgment received. If the transmit area is full when a datagram is forwarded to a LineConnector, then the datagram is dropped. The recovery for this datagram is up to the LIC-protocol.

In the RC5000 Network Switch a shared memory is used, and the communication between the processes is based on signal/wait of buffers.

To each LineConnector receive buffers from a common pool are signalled. When a datagram is received from the line, the buffer is signalled as a transmit buffer to the next LineConnector or signalled to a LocalConnector. If no buffers are queued at the LineConnector, due to all buffers staying in transmit queues, then LineConnector enters a busy situation, where it does not accept to receive datagrams from the line.

4. 5. Additional Services

Besides routing, the datagram network implements a number of services. These are described in this section.

4. 5. 1. Network Clock

In the RcPAX datagram network a synchronized network clock is maintained in the RC5000 MegaSwitch nodes. In each node a local derivation from the clock may be configured, so the network clock may be synchronized over different time zones. The synchronized network clock is useful in making it possible to compare timestamps received from different nodes. For example all LCP answers and LCP events are in the nodes timestamped with the synchronized clock.

To obtain a synchronized clock, one or more nodes in the network must be configured to hold a master clock. The clock in these nodes must be synchronized to the real clock by means of operator intervention.

The rest of the nodes in the network are configured as holding a slave clock. The clock in these nodes is automatically synchronized by the nearest master clock. It may be specified that it is only accepted to synchronize after a master clock at a given address hierarchy.

When there actually is no path to a master clock, the clocks will not remain synchronized, but may in time drift apart.

Attempts have been made to always at least initialize the clock, regardless of the topology of the network and the configuration of the clocks. The clocks may be initialized from the NMC. This is, however, not feasible for nodes initialized by reading requests from a winchester disc. Therefore, clocks are initialized from a neighbour node, too. The only prerequisite is that a neighbour node with an initialized clock exists, there need not be any path to a master clock.

Notification is made to the NMC when the difference in the time received from two different master clocks exceeds a threshold. This means that it is time to synchronize the master clocks to the real clock by operator intervention.

4. 5. 2. Cyclic Redundancy Code

All packets passing a line are, of cause, extended with a cyclic redundancy code (CRC) as prescribed in the X.25/2 LAPB protocol.

As a security check, and in order to encircle errors the NNP packets: Line Information, Range Vector, Network Clock and Broadcast, are containing a CRC. This CRC is computed and checked by the processes at each end of a line, i.e. only involving the passing of one line. The action on receiving NNP packets with CRC error may be configured from the NMC. The action may be set to reset the HDLC line, i.e. disconnect the HDLC and then reestablish the connection, or stop the line, i.e. the line is not used further without operator intervention.

An end-to-end CRC check may be calculated as a datagram is entering the datagram network and checked when the datagram is leaving the datagram network. This end-to-end check may, individually for each node, be configured on a per extension basis; that is, all datagrams originated from a given extension are filled in with the CRC and checked for the CRC before delivered to the addressed extension.

Another way of testing the quality of a path using the CRC, is to send Router-Router packets between the two end Routers.

4. 5. 3. Performance Management

Several statistics are counted in order to provide information for performance analysis.

A byte in the router header is used for containing a value only used for counting statistics; this value is called the stat type of the datagram. A datagram is, every time it is transmitted or received over a line, counted in the stat type statistics.

The stat type may, individually for each node, be configured on a per extension basis; that is, all datagrams originated from a given extension are assigned a configured stat type. The amount of transmission capacity used for a specific purpose, may be calculated on the basis of the stat type statistics.

Other stastistic counters are supported as well. E.g. the utilization of the lines may be calculated from the number of bytes received/transmitted and knowledge of the linespeed.

The amount of congestion in the network may be analyzed from the statistics of the number of routed and dropped datagrams.

The time the lines are accessible may be computed on the basis of the Router Connection LCP events.

4. 5. 4. Line Configuration

As a tool for controlling the configuration of lines in the network, the line identification is implemented.

Each line may be assigned a unique value identifying the line. The line identification is configured at the processes at both ends of the line. The configured line identification is exchanged over the line when a line is to be used as a router line, and a notification is given to the NMC if the identifications at the two ends are different.

4. 5. 5. Broadcast

The RcPAX datagram network implements broadcasting a datagram to all or a part of all nodes in the network. The datagram is routed to all processes with a specified extension address in (part of) the network. Only the RC5000 MegaSwitch nodes have implemented the broadcast algorithm.

The propagation of a broadcast may be controlled in different ways. The datagram may be broadcast to all nodes at the same address hierarchy (all nodes, all nodes with the same net address, all nodes with the same net and region address) or it may be broadcast a maximum number of hops (passing a line is counted as one hop).

Starting the broadcast of a datagram is done through the LCP interface of the Router process. The datagram is sent over all lines and is not protected by the LIC protocol. It is not possible to guarantee the delivery of the datagram to all nodes. If a copy of the datagram is dropped in the datagram network, no retransmission will occur, but the datagram will probably be delivered anyway by one of the other copies following other paths. The Router ensures that not more than one copy of the same datagram is delivered to the user.

No user of the datagram network which accepts to receive broadcast datagrams has yet been programmed.

4. 5. 6. Router-Router Packets

As a tool for implementing algorithms for test or analysis of the network, a protocol for sending packets from one Router process to another Router process is implemented. These packets are called Router-Router packets. Only the RC5000 MegaSwitch nodes have implemented the Router-Router algorithm.

Router-Router packets have the same router header as the usual datagrams switched in the network. They are distinguished on the extension address, which is unique to the Router process. The Router-Router packets are routed through the datagram network to the destination extension (a Router process) in the same way as all other datagrams are routed.

The sending of Router-Router packets is controlled through the LCP interface of the Router process. The Router-Router packets are not protected by the LIC protocol, and therefore they are not guaranteed to be delivered.

5. The Logical Internal Channel (LIC)

The datagram service offered by the ROUTER software component is by definition unreliable in the sense that datagrams may be lost, received out of sequence or even duplicated inside the datagram network.

To cope with these deficiencies the LIC software component implements a reliable, connection-oriented service. This connection service is established on top of the datagram service.

The connection service is in turn used by the software components implementing the access service, i.e. the DCE, STE and IDTE, for their internal communication. The DCE, STE and IDTE software components make use of the connection service, and are therefore called the LIC-users.

To each LIC-user entity, i.e. an activation of the DCE, STE or IDTE software component, corresponds exactly one LIC entity, which in turn is connected to a LocalConnector of the ROUTER entity. The LocalConnector can handle more than one LIC entity. It is therefore seen, that a LIC entity is engaged in two interfaces: on one side towards the LocalConnector and on the other side towards the LIC-user entity, see Fig. 36.

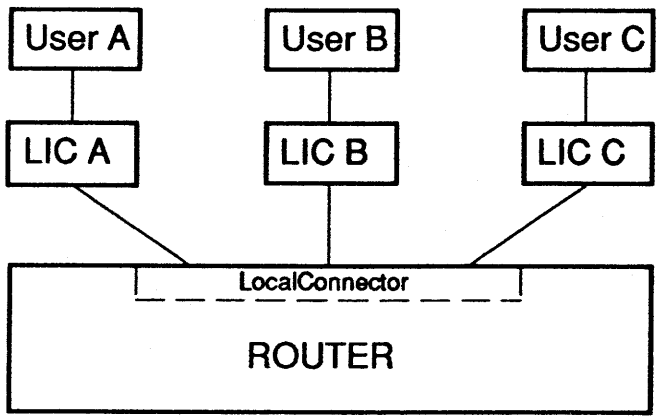


Figure 36 Entity Interfaces

Both the LIC-user/LIC interface (also just called the LIC interface) and the LIC/LocalConnector interface are internal software interfaces within the node.

Each node in an RcPAX network is uniquely identified by its net, region and node address within the RcPAX datagram address space of an RcPAX network. As each LIC entity identifies itself to a LocalConnector with its extension, and this extension is unique within the node, each LIC entity is uniquely identified within the RcPAX datagram address space of the RcPAX network in question by its RcPAX datagram address.

Due to the one-to-one relationship between a LIC entity and its LIC-user entity, this in fact implies that also all LIC-user entities, i.e. all DCE, STE and IDTE entities, are uniquely addressable within the RcPAX network's datagram address space, and that the LIC entity and the corresponding LIC-user entity "shares" the same datagram address.

5. 1. The LIC Services

The LIC services are provided on the LIC interfaces of two cooperating LIC entities running the internal LIC protocol on top of the datagram service. The LIC service is in turn made use of by the two corresponding LIC-user entities, see fig. 37.

The end-points of the connections on which the connection-oriented LIC services are provided on the LIC interface are termed LIC ports (LICPs). A LIC entity can handle a number of LICPs simultaneously, see fig. 38.

Each LICP of a LIC entity is identified by a set of two identifiers, one provided by the LIC entity and one provided by the LIC-user entity. The LIC and the LIC-user entities shall ensure that the LICP is unambiguously identified within the scope of the LIC interface in question.

These identifiers together with the unique RcPAX datagram address of the LIC entity, make the LICPs uniquely identifiable within a RcPAX network. It is seen that all LICPs on a LIC interface have a common datagram address which is that of the LIC entity supporting the LICPs.

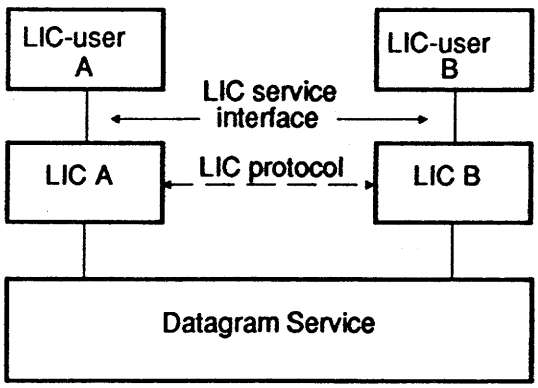


Figure 37 The LIC Service

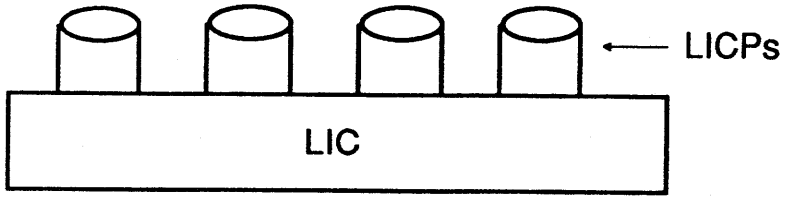


Figure 38 The LIC Ports (LICPs)

An association between a pair of two LICPs is denoted a logical internal channel (lic). Only a single lic can be handled between a pair of LICPs. The lic is identified by the combination of the identification of the two involved LICPs, called the source LICP and destination LICP respectively, see fig. 39. The source LICP is the LICP where the LIC establishment is initiated.

The LIC services are considered to consist of four services:

- the supervisory service
- the packet service
- the message service
- the datagram pass through service.

Supervisory service
The supervisory service is used to control logical internal channels. It includes services to establish, reset and remove logical internal channels.

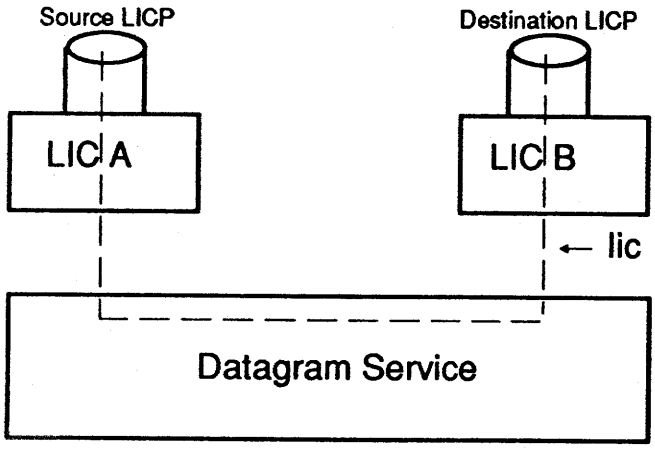


Figure 39 The Logical Internal Channel (lic)

Packet service
On an established lic, the packet service offers means of transfer of packets between the two involved LICPs. The service is full duplex meaning that the flow of packets in the two directions are handled independently of each other.

The packet service quarantees that packets are delivered and that they are delivered in the right sequence, i.e. the sequence of the sending LIC-user entity. The packet transfer is regulated by an end- to-end flow control mechanism (see further below).

In case the two involved LIC entities for some reason cannot live up to the guaranteed packet service, they initiate the removal of the lic in question, and thereby inform the LIC-user entities that the guarantees no longer hold.

Message service
The message service is also offered on an established LIC. Both the packet and the message service are offered simultaneously on the lics.

The message service offers delivery guarantee but no flow control.

Like the packet service, the message service is full duplex, i.e. messages are transferred independently in the two directions on a lic.

As for the packet service the LIC-user entities are informed through the removal of the lic if the guarantees of the message service cannot be lived up to.

Datagram pass through service
This service allows LIC-user entities to exchange datagrams between them using the LIC interface. No association between the LIC-user entities shall be established first, so datagrams are sent "outside" logical internal channels.

The LIC entities map this service directly onto the datagram service provided by the ROUTER software component, so no delivery, sequencing and non-duplication of datagrams are guaranteed.

Some elements of the packet and message services are described in the following.

Fragmentation/assembling
The LIC entity handles the packets and messages given by the LIC- user entity as single data units, implying that no fragmentation or assembling of long packets/messages takes place within the LIC entity. However, as the datagram service provides a fragmentation/assembling service which is used by the LIC entities, the packets and messages may have any appropriate size; fragmentation and assembling will take place as required in the LocalConnectors of the ROUTER entities involved.

Flow control
For the packet service a mechanism enabling the LIC-user entities to control the flow of packets on a lic is offered.

This flow control mechanism is based on a simple credit scheme: A LIC-user entity is not allowed to send a packet before it has obtained a credit from the other (potentially receiving) LIC-user entity. A credit can be seen as a token allowing one packet to be sent.

Each LIC-user entity issues a number of credits according to its current receiving capabilities. The issued credits are transported to the other LIC-user entity by the LIC entities involved, telling it how many packets it may send. For each packet the LIC-user entity sends, the number of available credits is decreased by one.

The rate of which a LIC-user issues new credits can depend on the number of receiving resources available, and can also be coupled to the flow control mechanism used for the further handling of the received packets. The DCE, for instance, couples the X.25 level 3 window flow control mechanism to the LIC credit mechanism, thereby implementing the end-to-end flow control required for the X.25 access service.

On a single lic a LIC-user entity is not allowed to issue more than a system dependent number of credits at a time (presently maximum 16 credits).

Although a LIC-user entity in principle is not allowed to send more packets than available credits, as an implementation optimization, it is allowed to do so. It shall then recognize that it has a negative number of credits. Packets sent by the LIC-user entity when it has no more (positive) credit will be queued inside the corresponding LIC entity, until appropriate credit is issued by the receiving LIC-user entity.

The transfer of credits between the LIC-user entities is independent of the packet transfer, i.e. even if packet transfer is blocked in one direction, credits for packet transfer in the opposite direction can be issued/delivered.

Reset
In some (failure) situations a LIC-user entity may require an established lic to be reset. A reset forces a lic and the transfer of packets and messages into a well defined state, from which the communication reliably can be resumed.

During reset pending packets and messages (i.e. in transfer between the two LIC-user entities) are dropped and the delivery control are suspended. This implies that the LIC-user entities have no knowledge of whether or not packets/messages sent and being in transfer at the initiation of a reset actually are delivered to the other LIC-user entity.

However, the packet and message services guarantee that packets/messages sent after a reset will not be delivered to the other LIC-user entity before that LIC-user entity has been informed of the reset. A reset can therefore be seen to form a synchronization point in the communication between the two LIC-user entities.

A reset is always initiated by one (or both) of the LIC-user entities, never by the LIC entities.

All credits are withdrawn during reset, i.e. to proceed packet transfer the LIC-user entities have to (re-)issue new credits.

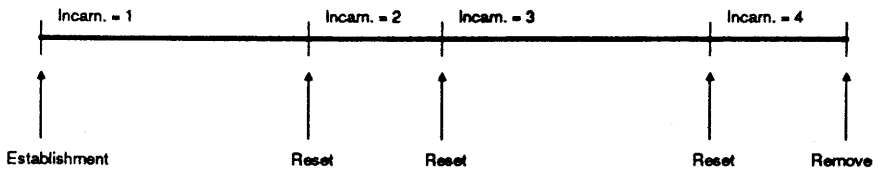


Figure 40 LIC Incarnations

Each time a lic is reset, it is said to come into a new incarnation. The incarnation is used to determine if a packet/message was sent before of after a reset. To identify the lic incarnations, the LIC entities manages an incarnation number for each lic. This number is increased for each reset.

5. 2. The LIC Service Interface

This section describes the LIC interface through which the LIC services as outlined in the preceeding section are provided.

The following description based on service primitives provides an overview of the flow of information between the involved LIC entities (the service-provider) and the LIC-user entities (the service-users) without going into implementation details.

The description follows the OSI style for service descriptions (see e.g. ISO/TR 8509) and is as such different than older and implementation oriented RcPAX documentation. A warning should therefore be given that such documentation can not necessarily be directly related to this new description style.

Before the service primitives are described, some comments on the implementation of the LIC service interface are provided as follows.

The implementation of the LIC interface is based on the mechanisms of software processes communicating by exchanging buffers via buffer queues.

This means that the LIC interface "by nature" is asynchronous implying that the LIC entity and the LIC-user entity may have different perceptions on the state of their interface at a given point in time. Therefore collisions may happen on the LIC interface, which have to be resolved. This is especially the case when resetting a lic.

All buffers on the LIC service interface are provided by the LIC-user entity. The buffers are said to be "signalled" from the LIC-user entity when handed over to the LIC entity and "returned" in the opposite direction.

In general two types of buffers are considered on the LIC interface:

Output buffers
These buffers are used to carry request and response primitives from the LIC-user entity to the LIC entity. An output buffer is pending at the LIC entity as long as the execution of the request/response primitive proceeds and then returned to the LIC-user entity, i.e. the buffers may be pending at the LIC entity for at rather long time. The LIC-entity puts some information indicating the result of the execution of the request/response in the returned buffer (and the reason in case of failure). This information can normally be ignored by the LIC-user entity.

The output buffers are used by the LIC entity to send internal LIC protocol data units to the remote LIC entity, i.e. the buffers may be signalled along to the ROUTER entity and be pending there while being handled.

Input buffers
Input buffers are used to carry indication and confirm primitives from the LIC entity to the LIC-user entity. An input buffer is signalled by the LIC-user entity to the LIC entity and will be pending until it is returned to the LIC-user entity carrying an indication or confirm primitive.

Two sets of input buffers exist, normal and local input buffers. The LIC entity signals the normal input buffers along to the ROUTER entity in order to receive internal LIC protocol data units sent by the remote LIC entity. A normal input buffer is pending at the ROUTER entity until a protocol data unit is received (an RcPAX datagram) and then returned to the LIC entity, which in turn may return the buffer issuing an indication or confirm primitive to the LIC-user entity.

It is the responsibility of the LIC-user entity to provide the required amount of normal input buffers. If it fails to do so, the ROUTER entity cannot deliver the datagrams to the LIC entity at the required rate. The ROUTER entity will then start dropping datagrams, which will trigger a number of recovery mechanisms in the internal LIC protocol. The overall performance will be degraded significantly.

Normal input buffers may be queued inside the LIC entity after their return from the ROUTER entity (e.g. packets received of out sequence). The LIC-user entity cannot therefore foresee lack of normal input buffers at the ROUTER entity. The primitivies LACK_OF_BUFFER (5.2.8) and SET_THRESHOLD (5.2.9) are used to cope with this situation.

Local input buffers are withheld by the LIC entity (not passed along to the ROUTER entity). The purpose of the local input buffers is to ensure that the LIC entity "always" has some buffers for indication and confirm primitives. It is therefore important that the LIC-user entity persistently delivers local input buffers to the LIC entity.

5. 2. 1. LIC establishment

The establishment of a lic is performed using the ESTABLISH primitive:

- the calling LIC-user entity issues an ESTABLISH request,
- this results in an ESTABLISH indication to the called LIC-user entity,
- the called LIC-user answers the ESTABLISH indication by issuing an ESTABLISH response,
- the ESTABLISH response results in an ESTABLISH confirm to the calling LIC-user entity.

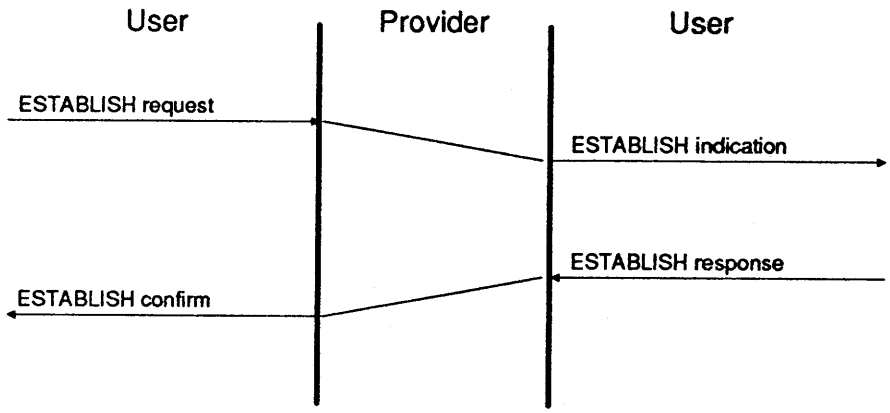


Figure 41 Lic Establishment

The LIC-user entities may start using the lic for packet and message transfer as soon as the lic is completely established for each of the LIC-user entities, that is:

- when the calling LIC-user entity receives the ESTABLISH confirm,
- when the called LIC-user entity issues the ESTABLISH response.

The lic establishment may fail for many reasons. In all phases of the lic establishment process both the LIC and LIC-user entities may initiate the removal of the partly established lic, see section 5.2.2.

An important case is when the called LIC-user entity for some reason does not want to establish the new lic, e.g. no more resources. In that case, the called LIC-user has to initiate the removal of the lic.

There exists no collision resolution in relation to lic establishment, so if the two LIC-user entities simultaneously initiate lic establishment to each other, two lics will (attempted to) be established.

5. 2. 2. Lic removal

The removal of a lic can be initiated either by the LIC-user entities or by the LIC entitites.

For LIC-user initiated removal:

- the removing LIC-user entity initiates the removal by issuing a REMOVE request,
- if the lic is at least partly established at the other LIC-user entity, i.e. it has received an ESTABLISH indication (and no removal has yet been initiated), a REMOVE indication is issued to it,
- when the lic has been removed, a REMOVE confirm is issued to the removing LIC-user entity.

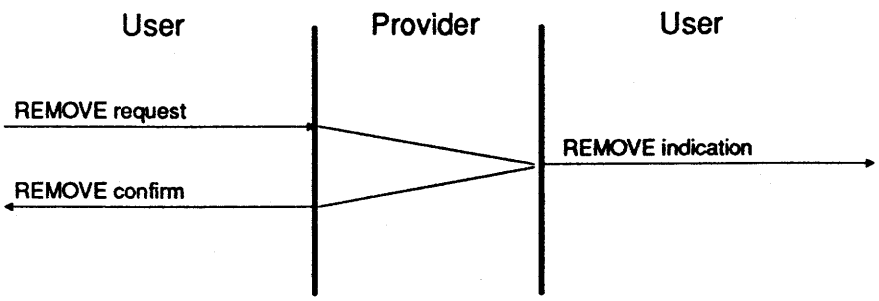


Figure 42 Lic Removal (LIC-user initiated)

For LIC initiated removal:

- the removing LIC entity issues a REMOVE indication to its local LIC-user entity,
- if the lic is at least partly established to the remote LIC-user entity, i.e. it has received an ESTABLISH indication (and no removal has yet been initiated), a REMOVE indication is issued to it.

A LIC-user entity shall consider the lic to be removed when it has received either a REMOVE confirm or REMOVE indication, i.e. it must not issue any primitive on this lic anymore.

Packets and messages in transfer on a lic may be lost during the remove, i.e. no delivery guarantee.

As all the involved entities are allowed to initiate the removal of the lic, a number of collision situations may occur. These are resolved by the LIC entities and the rule that

- a LIC-user entity that has issued a REMOVE request shall regard a received REMOVE indication as a REMOVE confirm.

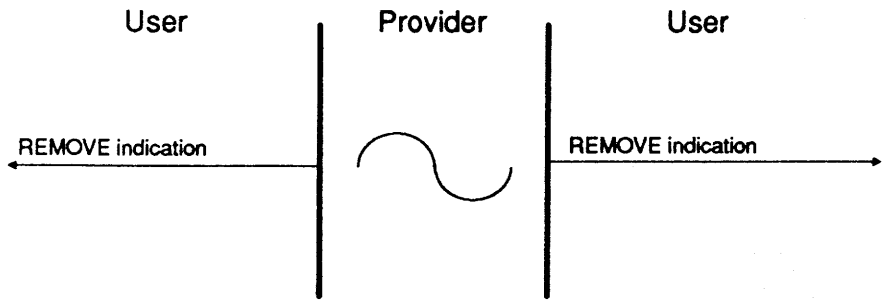


Figure 43 Lic Removal (LIC initiated)

In fact, the implementation of the LIC interface do not distinguish between the REMOVE confirm and indication primitives.

The LIC initiate removal procedure takes precedence of the LIC-user initiate procedure, so this may at any point be interrupted by the LIC initiated procedure.

If the LIC entities encounter any problems executing the LIC-user initiated remove procedure, the LIC initiated procedure is started.

5. 2. 3. Lic reset

The LIC-user entities may initiate a reset of an established lic, i.e. the establish procedure must have been completed and no remove procedure initiated:

- the resetting LIC-user entity issues a RESET request,
- this results in a RESET indication issued to the remote LIC-user entity,
- when the reset of the lic is finished, a RESET confirm is issued to the resetting LIC-user entity.

The LIC-user entity shall consider the lic to be reset when it receives the RESET indication or RESET confirm as appropriate.

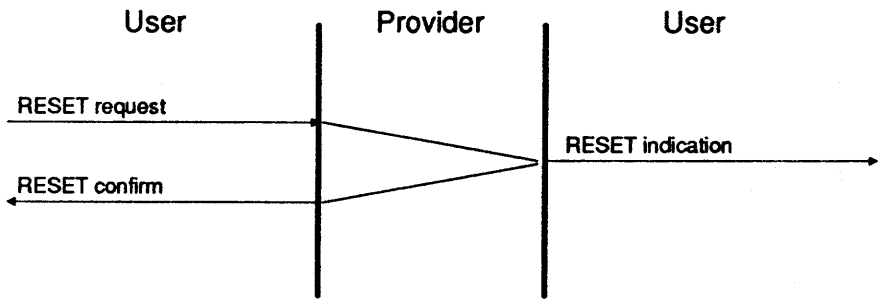


Figure 44 Lic Reset

The LIC-user entities do not know the delivery status of packets and messages sent before the reset was initiated, but after the reset the LIC-user entities receives only what the other LIC-user entity has sent after the reset.

All previously issued credits are withdrawn by the reset, i.e. the number of issued/available credits shall be set to zero. The resetting LIC-user may however issue new credits in the RESET request.

Both of the lic remove procedures take precedence of the lic reset procedure, so all of the involved entities may abort the reset procedure by starting a remove procedure, see 5.2.2.

Both the LIC-user entities may invoke the reset procedure simultaneously by issuing RESET requests. This collision is resolved by the LIC entities so both LIC-user entities will receive RESET confirm and no RESET indication as illustrated fig. 45.

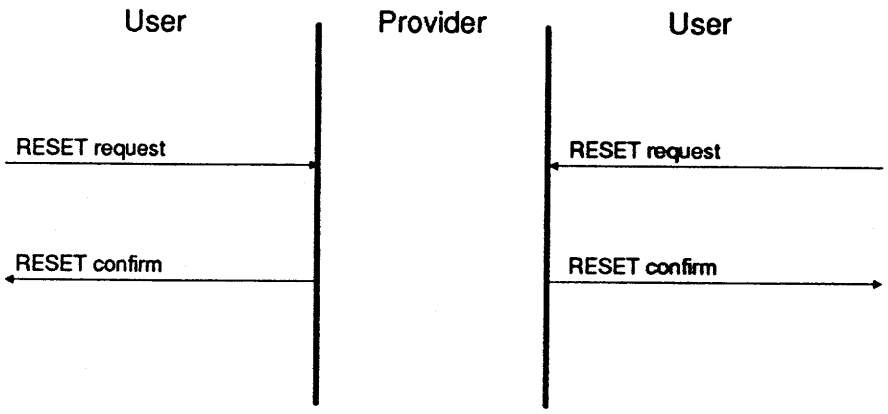


Figure 45 Lic Reset Collision

A reset collision may also take place between a LIC-user entity and the LIC entity, i.e. the LIC-user receives a RESET indication while waiting for a RESET confirm. In this case the LIC is reset twice, first reset corresponding to the RESET indication and then reset corresponding to the RESET request and related RESET confirm.

5. 2. 4. Packet transfer

The DATA primitives are used for transfer of packets between the two involved LIC-user entities:

- the sending LIC-user entity issues a DATA request,
- this results in a DATA indication issued to the receiving LIC-user entity.

If possible, the LIC-user entities will normally issue a number of DATA request to ensure a "smooth" flow of packets. DATA indications are issued in the same sequence as the corresponding DATA request. DATA requests which cannot be handled for the time being by the LIC entity to which they are issued, will be queued as appropriate.

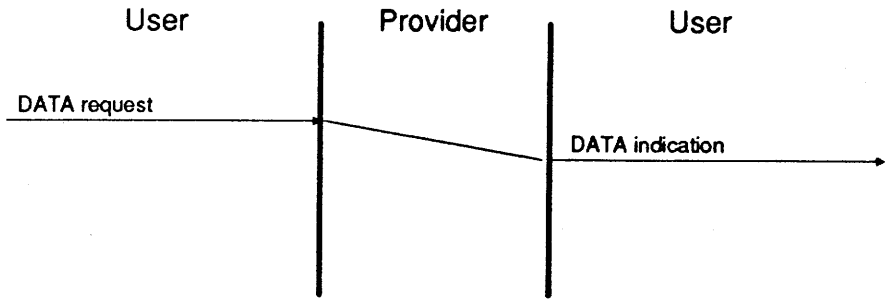


Figure 46 Packet Transfer

The remove and reset procedures take precedence of the packet transfer procedure. Previously issued DATA request may be lost during the remove or reset procedures.

The LIC-user may issue new credit in the DATA request primitive. The new credit will be delivered to the receiving LIC-user entity expeditely and independently of the delivery of the packet. This means that the new credit is not necessarily delivered in the DATA indication corresponding to the DATA request where the new credits were issued but may "overtake" it and be delivered in a DATA indication for a previously issued DATA request or even in an INCREASE CREDIT indication in case the packet transfer is blocked, see 5.2.5.

5. 2. 5. Increase credit

New credit for packets on established lics is handled using the INCREASE_CREDIT primitives, see fig. 47.

- The LIC-user entity issuing new credits issues an INCREASE_CREDIT request. The new credits allow the remote LIC- user entity to send more packets to the issuing LIC-user entity.

- This may result in an INCREASE_CREDIT indication to be issued to the remote LIC-user entity, or

- the new credit will alternatively be delivered implicitly in a DATA indication.

There is not a one-to-one relationship between one occurrance of an INCREASE_CREDIT request and one occurrance of an INCREASE_CREDIT indication or a DATA indication. Credits issued in one or more primitives (DATA or INCREASE_CREDIT) may be accumulated by the LIC entity and made available to the relevant LIC-user entity in one primitive (DATA or INCREASE_CREDIT).

The remove and reset procedures take precedence of the increase credit procedure. Credits issued before a reset are all withdrawn.

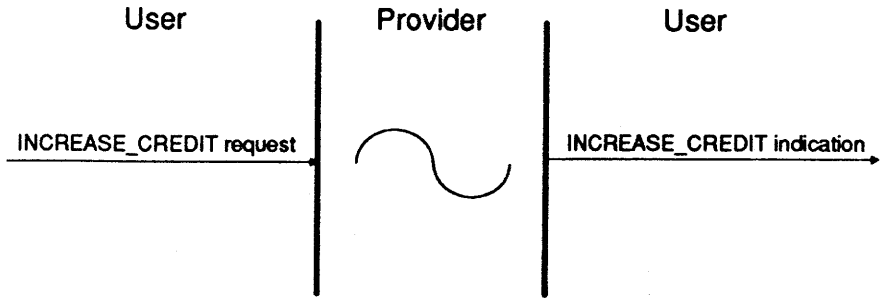


Figure 47 Increase Credit

5. 2. 6. Message transfer

Messages are handled on an established lic using the MESSAGE primitives:

- The sending LIC-user entity issues a MESSAGE request.
- This result in a MESSAGE indication issued to the receiving LIC- user entity.

A LIC entity will only execute one MESSAGE request at a time. If a LIC-user

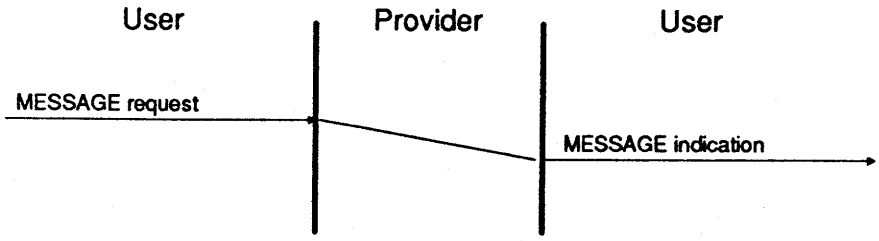


Figure 48 Message Transfer

entity issues more than one, the MESSAGE requests not being executed are queued inside the LIC entity receiving the MESSAGE request.

The LIC-user entity may specify that the transfer of packets in the same direction on the same lic shall be suspended while the message transfer takes place (hold).

As soon as the MESSAGE request has been executed, i.e. the MESSAGE indication has been issued, the packet transfer is resumed.

The remove and reset procedures take precedence at the message procedure, i.e. it may be interrupted at any time. Messages may be lost during the remove and reset procedures.

5. 2. 7. Datagram transfer

A LIC-user entity may request the LIC entity to transfer a datagram using the DATAGRAM primitives:

- The sending LIC-user entity issues a DATAGRAM request.
- This results in a DATAGRAM indication issued to the receiving LIC-user entity.

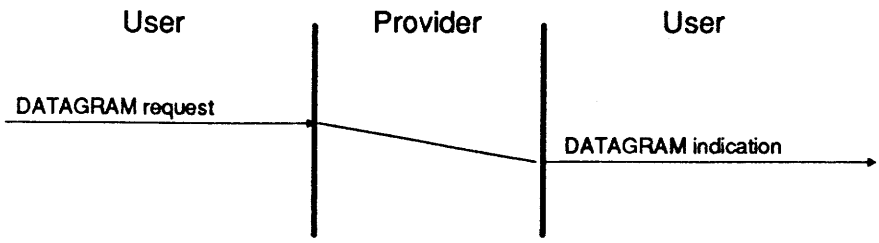


Figure 49 Datagram Transfer

The DATAGRAM indication may or may not be generated; there is no delivery guarantee.

This procedure is not influenced by any of the other LIC service procedures.

5. 2. 8. Buffer Lack

A threshold defines the number of normal input buffers that shall be pending at the ROUTER entity. In case the LIC entity detects that the actual number of normal input buffers are below this threshold it informs the LIC-user entity.

- The LIC entity issues a LACK_OF_BUFFER indication for each "missing" normal input buffer below the threshold.

The LACK_OF_BUFFER primitive takes no parameters.

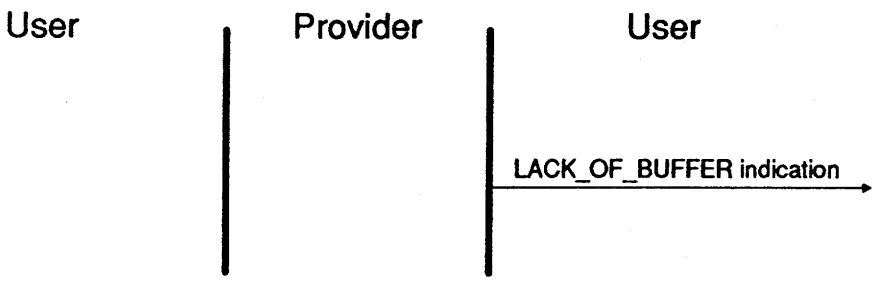


Figure 50 Buffer Lack

5. 2. 9. Set input buffer threshold

The LIC-user entity determines the threshold for the number of normal input buffers pending at the ROUTER entity:

- The LIC-user entity issues a SET_THRESHOLD request.

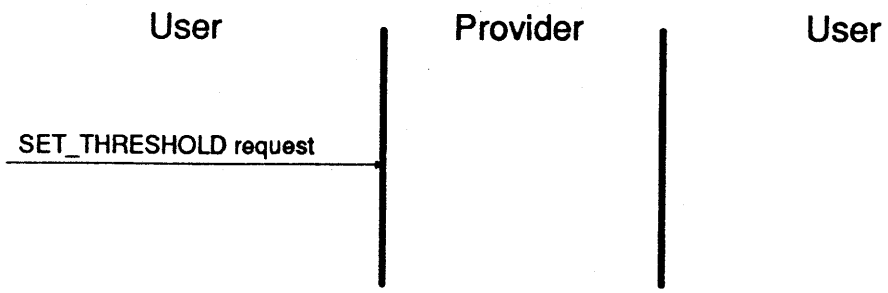


Figure 51 Set Threshold

The current value of the threshold determines when a LACK_OF_BUFFER indication is issued by the LIC entity, see 5.2.8.

The SET_THRESHOLD primitive carries a single parameter:

5. 3. The LIC protocol

This section provides an overview of the internal LIC protocol. It is not intended as a protocol specification as such.

The LIC entities cooperate by exchanging internal LIC protocol data units (LIC PDUs) in RcPAX datagrams using the datagram service provided by the ROUTER software component.

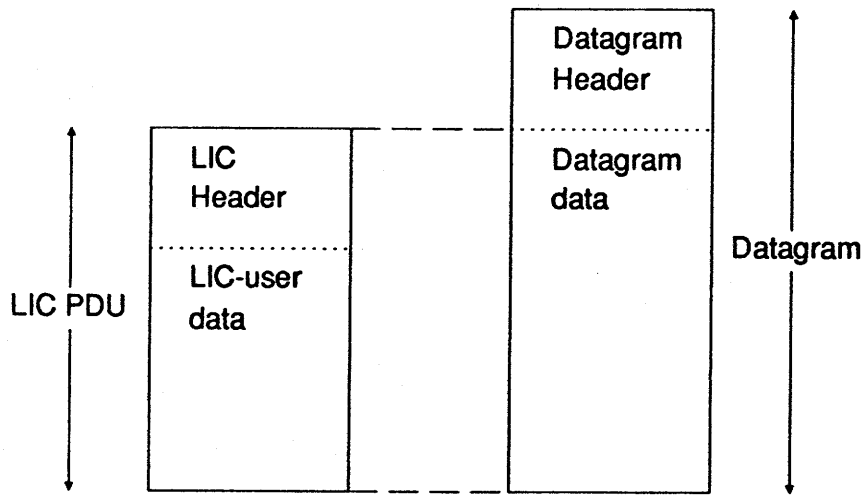


Figure 52 LIC Protocol Element

Each LIC PDU consists in general of a LIC header field and a field for LIC-user data, both contained in a datagram, see fig. 52 .

The LIC-user data field carries the data from the optional LIC-user parameters of the LIC service primitives described section 5.2, i.e. it may or may not be present in the LIC PDU. Not all LIC PDU types can have LIC-user data associated with it.

5. 3. 1. LIC header

The LIC header is composed of a fixed number of fields, i.e. they are all present in all LIC PDUs, see fig. 53 .

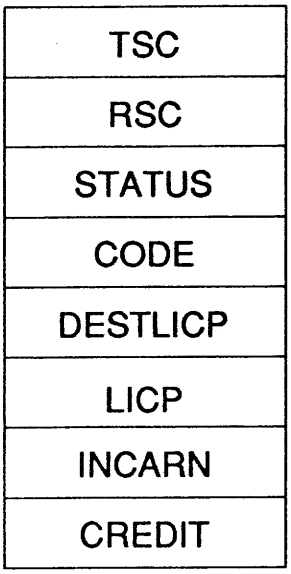


Figure 53 The LIC Header

DESTLICP, LICP
These fields carry the LIC ident of LIC to which the PDU is associated. The LICP field holds the LIC ident allocated by the LIC entity sending the PDU while the DESTLICP field is used for the LIC ident allocated by the LIC entity receiving the PDU.

CODE
This field indicates the type of LIC PDU. Fig. 54 provides an overview of the different types of PDUs and their usage.

TSC
This TSC (Transmit Sequence Count) field is used as a sequence number to identify each DATA_PACK and MSG_PACK PDUs sent over a lic. The sequence numbering of the two types of PDUs are independent, i.e. two sets of sequence numbers.

The TSC field is only used for DATA_PACK and MSG_PACK PDSUs, i.e. its value is undefined for all other PDUs.

PDU Type	USAGE
CALL	The CALL PDU is used to carry an ESTABLISH request from the calling LIC entity to called LIC entity.
CALL_ACK	Used by the called LIC entity to acknowledge the receipt of a CALL PDU.
CONF_CALL	The called LIC-entity uses the PDU to carry an ESTABLISH confirm back to the calling LIC entity.
CONF_CALL_ACK	Used by the calling LIC entity to acknowledge the receipt of a CONF_CALL.
CLEAR	This PDU is used to remove the lic, i.e. it carries a REMOVE primitive.
CLEAR_ACK	Used by a LIC entity to acknowledge the receipt of a CLEAR PDU.
RESET	This PDU is used to carry a RESET request from the setting LIC entity to the remote LIC entity.
RESET_ACK	The RESET_ACK PDU is used to acknowledge the receipt of a RESETPDU.
DATA_PACK	This PDU carries primarily the packets being transferred, i.e. the DATA primitives. New credits and acknowledgements for previously received DATA_PACK PDUs may, however, be piggybacked in a DATA_PACKPDU.
CRED_PACK	The CRED_PACK PDU is used to carry new credits and/or acknowledgements for previously received DATA_PACK PDU when they cannot be piggybacked in DATA_PACK PDUs.
MSG_PACK	Carries a message, i.e. corresponds to a MESSAGE request.
MSG_PACK_ACK	These PDUs are used to acknowledge the receipt of a MSG_PACK PDU.
DG_PACK	The DG_PACK PDU carries a datagram for the datagram pass through service, i.e. corresponds to the DATAGRAM request.

Figure 54 LIC PDUs

RSC
This RSC (Receive Sequence Count) field is used to acknowledge the receipt (by the sender of the PDU containing the RSC) of a DATA_PACK or MSG_PACK PDU.

In a DATA_PACK or CRED_PACK PDU the RSC value acknowledges all received DATA_PACK PDUs not yet acknowledged with TSC up to but not including the RSC value.

In a MSG_PACK_ACK PDU the RSC value identifies the TSC of the MSG_PACK PDU being acknowledged.

The RSC field is only used in DATA_PACK, CRED_PACK and MSG_PACK_ACK PDUs, i.e. its value is undefined for all other PDUs.

CREDIT
This field is used by the sender of the PDU to indicate the amount of credit currently issued, i.e. the number of DATA_PACK PDUs the receiver of the PDU may send.

The receiver of the PDU may send DATA_PACK PDUs with TSC values from RSC to RSC + CREDIT - 1.

The CREDIT field is used as described above in DATA_PACK and CRED_PACK PDUs.

For CALL, CONF_CALL and RESET PDUs the CREDIT field carries the initial credit for the lic after the stablishment/reset. The value in CREDIT field is in these cases not "relative" to the RSC field (which is undefined for these PDUs).

The value of the CREDIT field is undefined in the PDUs not mentioned above.

INCARN
This field carries the incarnation number of the lic to which the PDU belongs.

The INCARN field is used for all PDUs except the DG_PACK PDU for which its value is undefined.

STATUS
This field is used by the LIC entities to inform each other of the reason for sending this PDU, e.g. retransmitted due to missing acknowledgement. The action of the LIC entity receiving the PDU may depend on the reason for sending the PDU, i.e. the value of the STATUS field.

The STATUS field is used for all PDUs except the DG_PACK PDU for which its value is undefined.

5. 3. 2. LIC protocol operation

The operation of the LIC protocol is outlined in this section by a number of figures which illustrate the normal operation of the internal LIC protocol providing the LIC services. Only the most normal situations are shown; especially no recovery situations.

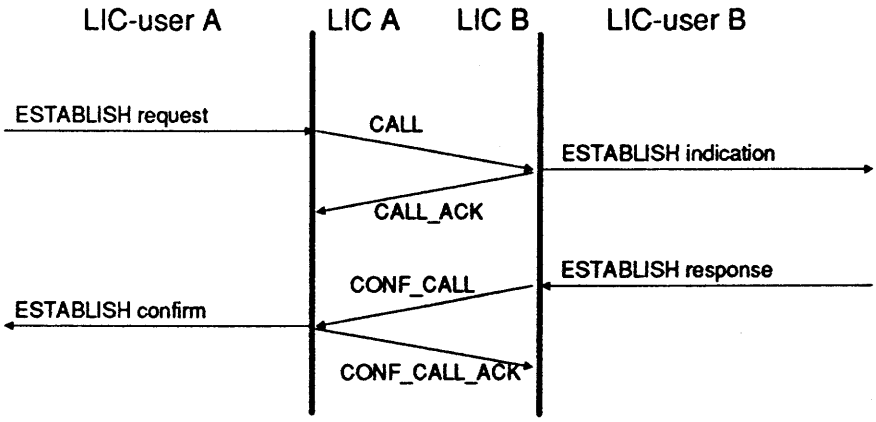


Figure 55 Establishment

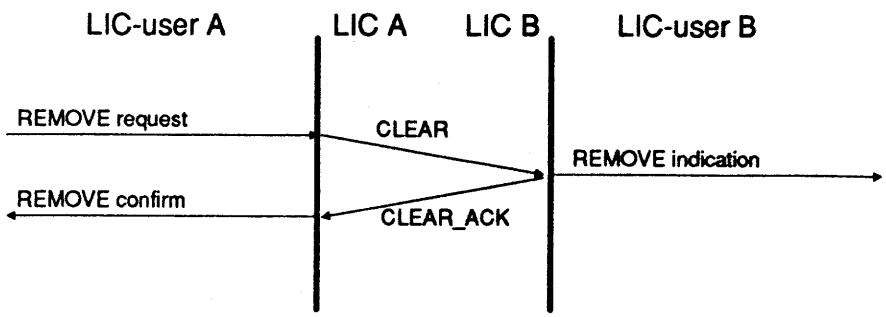


Figure 56 LIC-user Initiated Removal

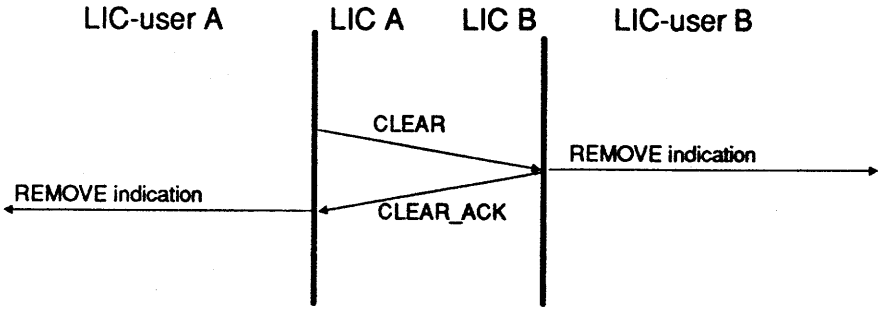


Figure 57 LIC Initiated Removal

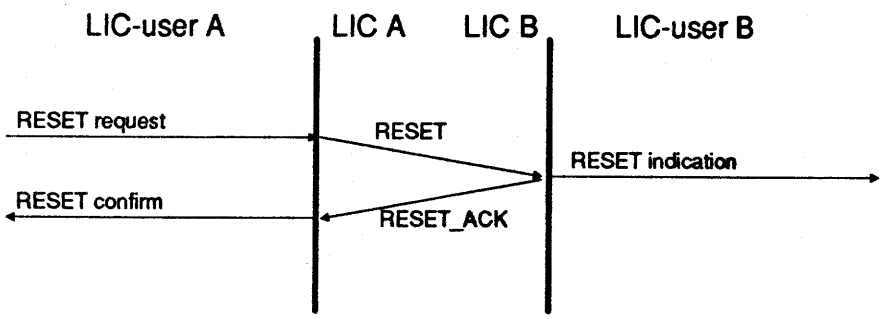


Figure 58 Reset

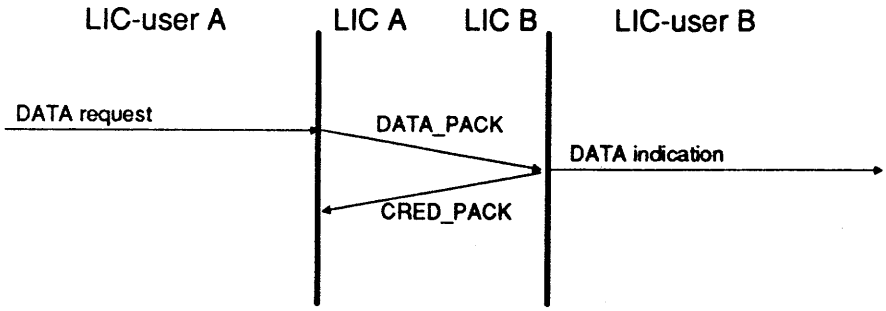


Figure 60 Packet Transfer

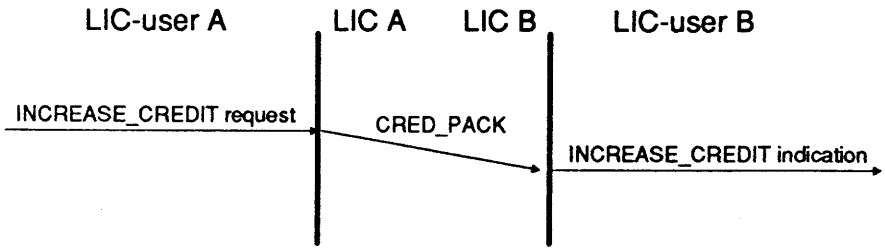


Figure 61 Increase Credit

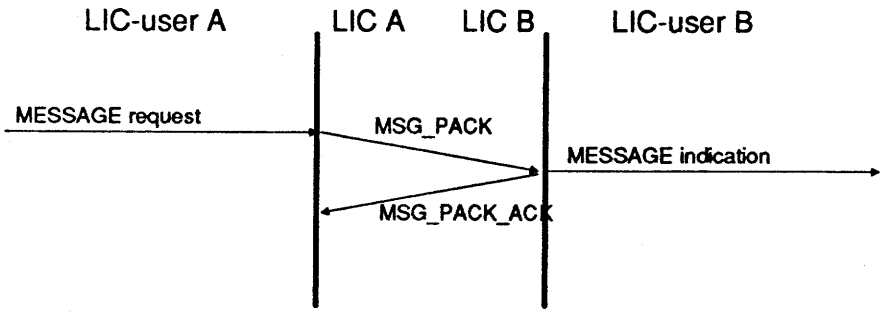


Figure 62 Message Transfer

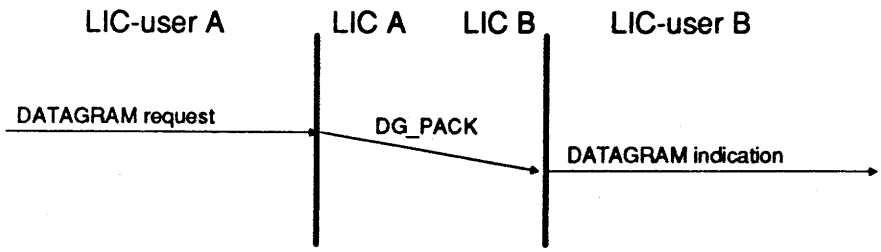


Figure 63 Datagram Transfer

5. 4. LIC Functions

To ensure the correct provision of the LIC services and the proper operation of the internal LIC protocol, some functionalities are required of the LIC software components additional to those mentioned hitherto.

5. 4. 1. Retransmission

In order to overcome loss of datagrams, the LIC entity implements a retransmission mechanism.

When issuing a request/response primitive, the LIC-user entity provides where relevant a Timeout value to the LIC-entity indicating how long time it is allowed to use for the execution of the request/response in question, i.e. make the related indication primitive happen.

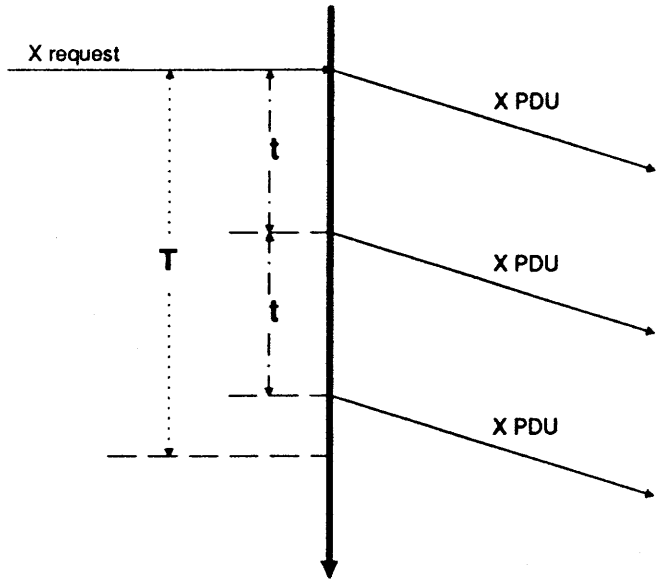
The general rule is that only PDUs carrying requests or responses may be retransmitted (CALL, CONF_CALL, CLEAR, RESET, DATA_PACK, MSG_PACK) and not the acknowledge PDUs (CALL_ACK, CONF_CALL_ACK, CLEAR_ACK, RESET_ACK, CRED_PACK, MSG_PACK_ACK) nor the DG_PACK PDU.

When a PDU has been sent, the LIC entity starts a retransmission timer. The PDU is retransmitted if the retransmission timer runs out before an acknowledge for the PDU has been received. The PDU is retransmitted as long as the Timeout value given by the LIC-user entity for the related request/response is not exceeded.

In each retransmitted PDU the LIC entity sets the STATUS field in the LIC header to indicate that the PDU has been retransmitted so the receiving LIC-entity can recognize a retransmitted PDU.

Usually the retransmission timer is much smaller than the LIC-user entity's Timeout value, so the LIC entity retransmits the PDU several times before it abandons the execution of the request/response and starts the LIC initiated remove procedure.

As the network topology influences the optimum value of the retransmission timer it should be determined at system configuration.



t: LIC retransmission timer
T: LIC-User Timeout value

Figure 64 Retransmission

In case no acknowledge is received for a CLEAR PDU, i.e. no CLEAR_ACK within the Timeout value, the LIC entity will remove its part of the LIC anyway.

Transmitted DATA_PACK PDUs which have not yet been acknowledged by the receiving LIC entity are set up in a retransmission queue within the sending LIC entity. The buffers containing the DATA_PACK PDUs are returned to the LIC-user entity from the retransmission queue as soon as they have been acknowledged by the receiving LIC- entity.

If the retransmission timer runs out, only the first DATA_PACK PDU in the retransmission queue is retransmitted, i.e. the "oldest" DATA_PACK PDU which not yet has been acknowledged.

This DATA_PACK PDU will be retransmitted until it is acknowledged or the Timeout value for the corresponding DATA request is exceeded.

5. 4. 2. Resequencing

DATA indications shall be issued to the receiving LIC-user entity in the same sequence as the corresponding DATA requests were issued by the sending LIC-user entity.

As the DATA_PACK PDUs may arrive out of sequence the receiving LIC entity has to "sort them out" again. To this end a resequence queue is implemented where DATA_PACK PDUs received "too early", i.e. do not carry the next DATA indication to be issued to the received LIC-user entity, are withheld.

The DATA_PACK PDUs are taken out of the resequence queue as soon as the related DATA indications can be issued to the receiving LIC- user entity in the right order.

The receiving LIC entity acknowledges only those DATA_PACK PDUs for which the DATA indication has been issued. In case a DATA_PACK PDU is lost, the receiving LIC entity acknowledges only "up to" the lost DATA_PACK PDU, thereby making the sending LIC entity retransmit the lost DATA_PACK PDU.

5. 4. 3. PDU Integrity

As PDUs may pop up more than once at the receiving LIC entity (due to duplication in the datagram service or retransmission by the sending LIC entity) the LIC entity has to make sure that the received PDU is relevant for the lic in question.

The primary mechanism for this is the usage of the incarnation number. For all PDUs where the INCARN field of the LIC header is used, the value of the INCARN field shall match the current incarnation number of the lic before a received PDU is considered acceptable for the lic.

The LIC entity also maintains a comprehensive state machine to control the operation of the internal LIC protocol. This ensures that a received PDU is only acceptable if that type of PDU is expected (according to the internal LIC protocol) to arrive in the current state.

For received DATA_PACK PDUs it is an additional requirement that the TSC of the received PDU shall be within the interval from RSC to RSC + CREDIT - 1, i.e. the TSCs which have been "authorized" by the receiving LIC entity.

The TSC of a received MSG_PACK PDU shall be the next of the message sequence numbers, and the RSC of a received MSG_PACK_ACK PDU must match exactly the RSC of the unacknowledged MSG_PACK PDU.

5. 4. 4. Heartbeat

In case that no traffic is running on an established lic, a LIC entity cannot detect whether or not the remote LIC entity is still "alive". This problem is coped with as follows.

Unless no other PDUs are sent on a lic, the LIC entity sends CREDIT_PACK PDUs now and then to tell the other LIC entity that it still is "alive". The interval between the CREDIT_PACK PDUs is short just after a "normal" PDU has been sent and increases up to a certain limit for each "heartbeat" CREDIT_PACK PDU sent.

A LIC entity supervises that it receives at least the "heartbeat" CREDIT_PACK PDUs. If it does not receive any PDU within a given time period (allowing for CREDIT_PACK PDUs to be lost) it assumes that the other LIC entity is "dead" for some reason, and starts the LIC initiated remove procedure.

5. 4. 5. Cancel resequence queue

The LIC entity may in extreme situations lock up a considerable number of normal input buffers in the resequence queues, so many that the LIC-user entity cannot deliver more buffers allowing new PDUs to be received, see section 5.2.

To overcome this potential deadlock situation, the LIC entity starts a resequence queue timer when it issues a LACK_OF_BUFFER indication to the LIC-user entity. If the LIC entity has not received any new normal input buffers from the LIC-user entity when the resequence queue timer runs out, the LIC entity cancels its resequence queues.

When cancelling the resequence queues, all buffers held in the resequence queues are returned "empty" to the LIC-user entity allowing it to determine where its buffers are mostly needed (as the LIC-user entity can be in need for buffers for other purposes).

Note that all resequence queues are cancelled at the same time, so this affect potentially all the currently established lics for the LIC entity in question.

6. DCE and STE

This chapter describes both the X.25 software component and the X.75 software component. The software components are referred to as the DCE or the STE, respectively. Both software components implement the 1980- and 1984-recommendations. It is assumed that the reader is familiar with the recommendations (ref. 6,7).

In most areas the software components are very similar, which justifies the common treatment. In cases where differences exist it is explicitly noted in the text.

In the following sections the services, functions, protocols and interfaces of the software modules are described.

6. 1. General overview

The main task of the DCE and the STE software components is to implement an access interface to RcPAX in accordance with the CCITT X.25 and X.75 level 3 recommendations as described in ref. 6 and 7.

Fig. 65 illustrates how a DCE via a proprietary internal X.25 protocol communicates with an STE. In this case the 2 DTE's are in different networks (only the DTE connected to RcPAX is shown here). In fig. 69 the call between F and H shows this situation.

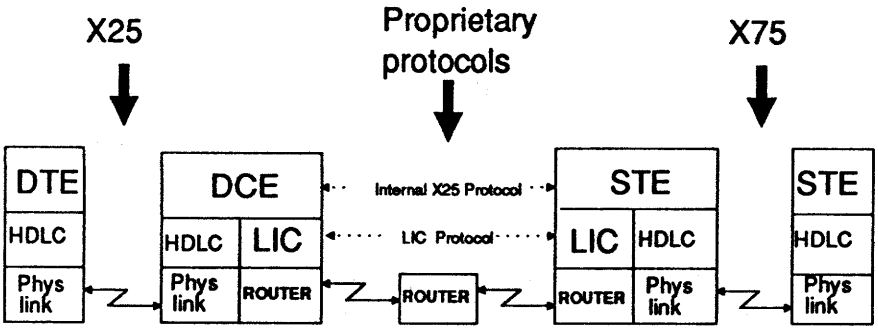


Figure 65 DCE Entity communicating with an STE Entity

The 2 other cases are

- 2 DCE's communicate. This means that both DTE's are in RcPAX.
- 2 STE's communicate. RcPAX is used for transit.

Fig. 65 shows further that the DCE/STE builds upon the services offered by the LIC software component, as described in chapter 5. The LIC uses the datagram service, and the term "lic" refers to the logical internal channels used to implement an end to end controlled transfer of packets through the network.

Fig. 66 shows another possible communication partner for the DCE/STE: the Internal DTE (IDTE), which is described in chapter 7. Taken together, fig. 65 and 66 show that DCE's, STE's and IDTE's communicate with each other by using the RcPAX Internal X.25 Protocol.

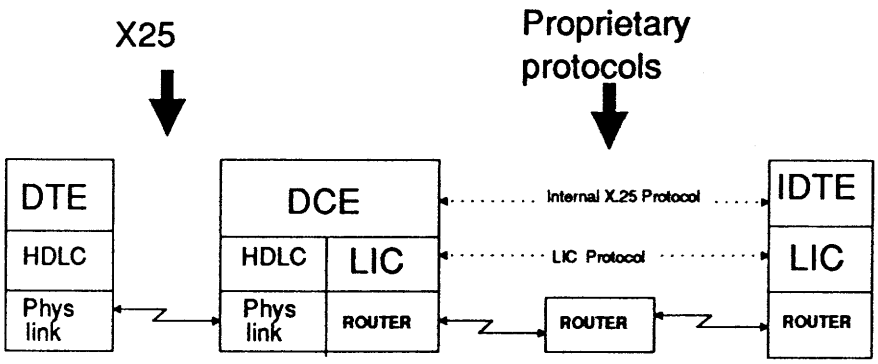


Figure 66 DCE Entity communicating with an IDTE Entity

The X.25/3 recommendation specifies the establishment, maintenance and use of Virtual Calls and Permanent Virtual Circuits between the DTE's connected to the network. In X.75 the specification concerns the interface between public networks through STE's, but the 2 recommendations are essentially very similar. In both cases the Virtual Call or Permanent Virtual Circuit is extended through the network by the DCE/STE to which the DTE/STE is physically connected. Fig. 67 illustrates how 2 X.25/3 logical channels are concatenated by one lic, together constituting a Virtual Call or Permanent Virtual Circuit.

This bonding of X.25/X.75 logical channels and logical internal channels is controlled by the DCE/STE software components in the associated access points of the network (access nodes). The DCE/STE has the initiative for the establishment and removal of lic's. The control is performed by carefully maintaining the integrity of individual channels. In this way each lic may be used to provide the DCE/STE with a tool for establishing a full-duplex packet transfer with:

- Flow Control
- Sequence Control
- Delivery Control

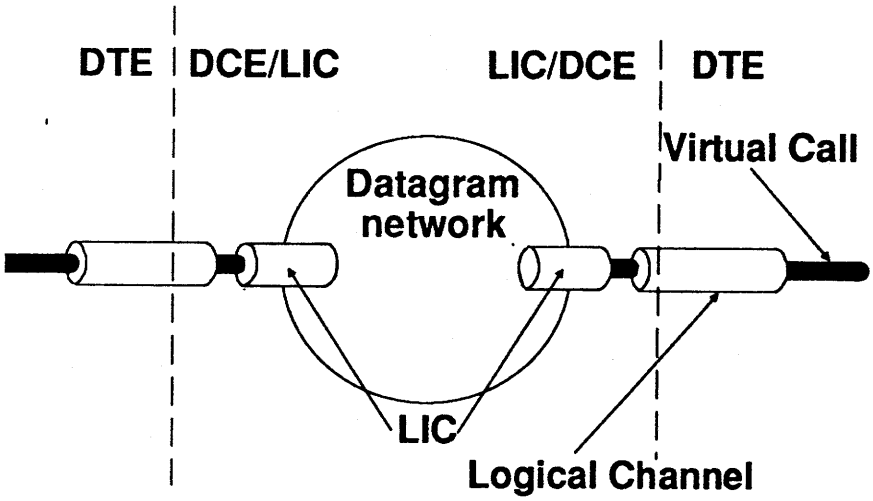


Figure 67 Concatenation of Channels

In implementing the X.25/X.75 access interface, the DCE/STE utilizes the services provided by the LIC software component for the transportation of packets. Thus the DCE/STE is relieved of the tasks forced by the inherent properties of the Datagram Network. As a consequence the design benefits from all the advantages of a layered architecture. This implies for instance that only the DCE/STE software component needs to be modified in order to conform to future revisions of the X.25/3 recommendation. Furthermore the actual transport mechanism may be changed without consequences for the DCE/STE software component.

6. 2. Services offered by the DCE/STE

Generally, the DCE/STE offers an X.25(X.75) level 3 service to the DTE/STE. In excess of that a number of additional services offered by RcPAX are described in this section. Finally, each supported user facility and network utility is listed.

6. 2. 1. Increased line utilization facility

The DCE/STE can be configured to offer an increased line utilization. An X.25(X.75)/3 category A packet can be concatenated with the following packet and sent through the DG net as one packet. This reduces the overhead (i.e. the ratio between control data and user data in the DG net packet) and thereby increases the possible line utilization of the access line.

6. 2. 2. 80/84 Interworking

Communicating partners in RcPAX - such as the DCE labelled A and the DCE labelled B in fig. 68 - must be able to exchange their versions during call setup, because the 1980 recommendation only allows a subset of the facilities defined in the 1984 recommendation, meaning that a 1980 DTE/STE must be protected from receiving facilities or utilities it does not understand.

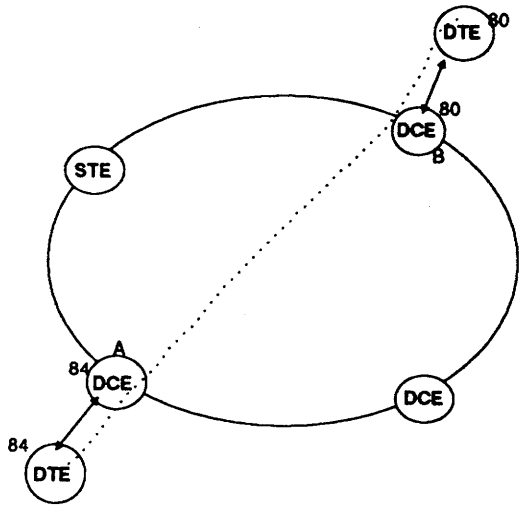


Figure 68 Call between 80 DCE and 84 DTE

During configuration the DCE/STE must be told whether to implement a 1980 or a 1984 version of the recommendation. In this way the DCE/STE is set up to "match" the version of the adjacent DTE/STE. In fig. 68 the DCE labelled A implements X.25 version 84 while the DCE labelled B implements X.25 version 80. The communicating partners in RcPAX (A and B) can be DCE-DCE, STE-STE or DCE-STE.

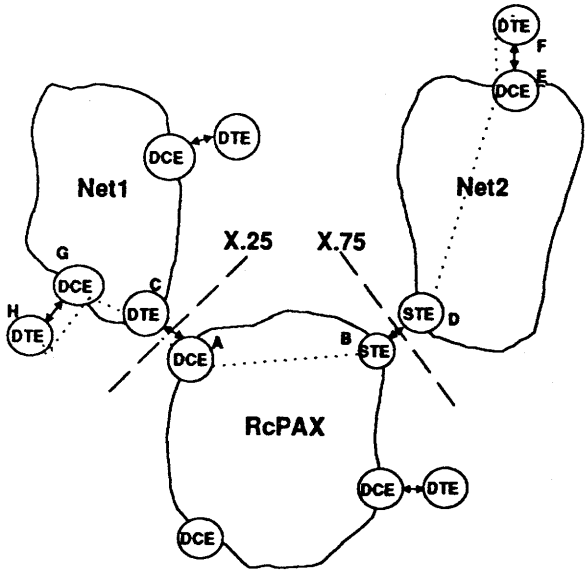


Figure 69 Transit Call

In fig. 68 both DTE's are directly connected to RcPAX through DCE's. Fig. 69 shows a situation, where 2 networks are connected to RcPAX: one through a DCE and the other through an STE. In this case the DCE labelled A will implement the X.25 version given by the DTE labelled C, and the STE labelled B will implement the same X.75 version as D.

Both node A and B are access points to RcPAX, and if address translation is used, a PNIC (Private Network Identification Code) is assigned to each connected network. The combination of the PNIC and the address in the private network uniquely identifies a DTE.

The RcPAX Internal X.25 Protocol (Fig. 65) allows communicating partners to exchange their versions during call setup.

6. 2. 3. RR-Timer

The RR-Timer can be used as a kind of "heartbeat process". When the timer runs out, the DCE/STE will send an RR-packet to the DTE/STE and then restart the timer. The timer can be disabled by setting its timer value to zero.

6. 2. 4. Address Translation Service

The DCE/STE can be used with or without logical addressing. If used without logical addressing, the DCE/STE will use physical (net, region, node) X.121 addressing.

If the creation request set of the DCE/STE contains a "Init address translation" request, logical addressing will be used, and the DCE/STE will call upon the services of the ATS to translate addresses and obtain the physical address of the destination node.

6. 2. 5. NUI validation

Primarily for use by the X.28 user, RcPAX offers the Network User ID Validation. The NUI user facility consists of 16 characters, where the first 8 characters are chosen by the administration while the last 8 characters are chosen by the user.

If a NUI-facility is found in a call request packet, the 16-character value is looked up in the DIS. The call is cleared unless the query is successful.

6. 2. 6. Call/Data-timers

Besides the 4 timers (T10-T13 for the DCE and T30-T33 for the STE) a number of timers (6) are implemented in order to monitor call setup and data exchange. These timers are:

Call Request Timer
Clear Request Timer
Reset Request Timer
Accept Incoming Call Timer
Interrupt Timer
Data Timer

The job of the timers are to ensure that a call is cleared, if the remote end does not answer. A more detailed description of the timers can be found in ref. 8.

6. 2. 7. X.2 Facilities supported by the DCE

Extended packet sequence numbering (mod 128).

Nonstandard default window sizes.

Nonstandard default packet sizes.

Default throughput classes assignment.

Flow control parameter negotiation.

Throughput class negotiation.

Packet retransmission.

Incoming calls barred.

Outgoing calls barred.

One-way logical channel outgoing .

One-way logical channel incoming.

Closed user group.

Closed user group with outgoing access.

Closed user group with incoming access.

Incoming calls barred within a closed user group.

Outgoing calls barred within a closed user group.

Bilateral closed user group.

Bilateral closed user group with outgoing access.

Reverse charging acceptance.

Fast select acceptance.

Charging information.

Direct call.

Hunt group.

D-bit modification.

Local charging prevention.

Call redirection.

Network user identification.

Extended frame sequence numbering.

Closed user group selection.

Bilateral closed user group selection.

Reverse charging.

Flow control parameter negotiation.

Fast select.

Throughput class negotiation.

Charging information.

Call redirection notification.

Called line address modified notification.

Network user identification.

Closed user group with outgoing access selection.

Transit delay selection and indication

The network accepts the transit delay selection facility, but does not in any way react upon it. The transit delay indication facility provides for one value of the transit delay per DCE. This value can be set from the NMC.

Address extension

The CCITT specified DTE facilities to support the OSI Network Service (Calling- and called-address extension) as specified in Annex G of the X.25 recommendation are implemented in both the DCE and the STE. In case of calling address extension the NSAP-address in the call request packet is validated, and in case of called address extension the NSAP-address is looked up by the address translation service to obtain the router address.

6. 2. 8. X.75 Network utilities supported by the STE

Transit network identification.

Call identifier.

Throughput class indication.

Window size indication.

Packet size indication.

Fast select indication.

Closed user group indication.

Closed user group with outgoing access indication.

Reverse charging indication.

Called line addres modified notification.

Clearing network identification code.

Utility marker.

Address extension.

The STE will act upon an address extension utility in the same way the DCE acts on the address extension facility.

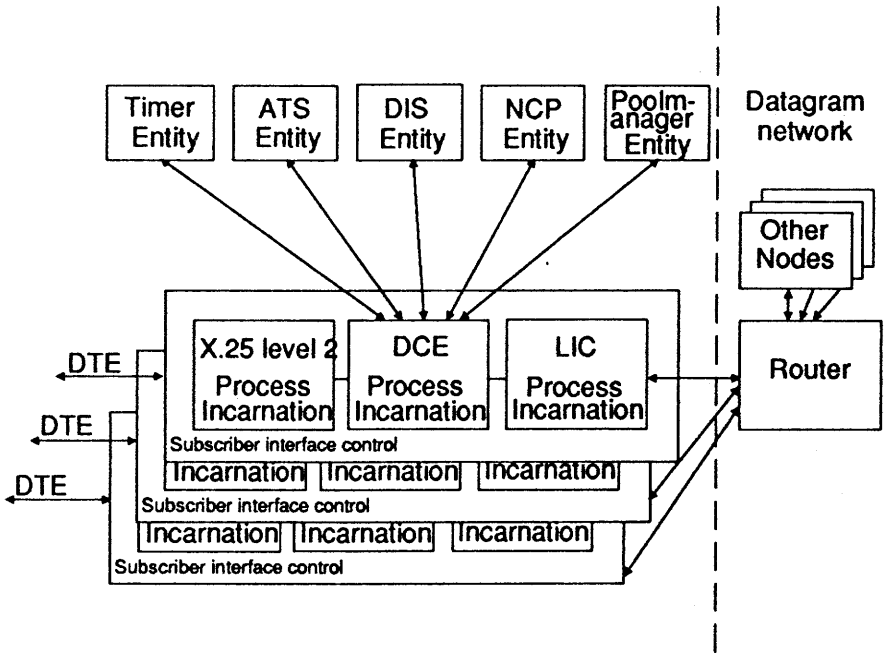


Figure 70 Software Structure in Access Node

6. 3. Services used by the DCE/STE

In order to service its user the DCE/STE draws upon the service offered by several software components in the node. One may consider that there to every DTE connected to the network is associated a "logical cluster" of entities intended to service the DTE, and to provide an interface towards the DG net. This software structure is shown in fig. 70.

Note that although the DCE/STE uses the services of the LIC as mentioned in the previous section, the figure emphasizes the fact that one LIC entity exists for each DCE/STE, while one entity of each of the other software components (including Timer, ATS, DIS, NCP and PoolManager) services all DCE/STE's in the node.

The DCE and the STE are each implemented as one RTP-process.

The HDLC-service is used by the DCE when exchanging packets with the DTE, and by the STE when exchanging packets with another STE, i.e. the level 3 packets are multiplexed in level 2 frames.

The Timer-service is used by the DCE/STE to implement the 4 timers T10-T13 (T30-T33 for the STE) as specified in the X.25 (X.75) recommendation, together with various internal timers.

The ATS-service is used by the DCE/STE to translate between logical and private addresses.

The DIS-service is used by the DCE/STE to validate DNIC's and NUI's and translate between Internal Cug Number (ICN) and International Interlock Code (IIC).

The NCP-service is used by the DCE/STE to report to the NMC.

The Poolmanager-service is used by the DCE/STE to manage buffer resources. As the name indicates, the Poolmanager only manages the buffers. Buffers are allocated by the DCE/STE and are explicitly given to the Poolmanager.

6. 4. External Interfaces

The interface between the DCE/STE and the software components servicing it (ATS, DIS, Timer, NCP, LIC), is basically a signal-wait programming interface.

In the following the interface to the LIC is described in some detail by showing how the X.25/X.75 functions are mapped onto the LIC primitives.

Level 3 function		Mapped onto
Call Request	--->	Establish lic
Incoming Call	< --	Establish Indication
Call Accepted	--->	Confirm establish
Call Connected	< ---	Confirm establish indication
Clear Request	--->	Remove lic
Clear Indication	< ---	Indication remove lic
Reset Request	--->	Reset lic
Reset Indication	< ---	Indication reset lic
Data	--->	Data
Interrupt	< ---	Send Message

As a LIC-user, the DCE/STE uses the Increase Credit LIC-primitive to issue credits to the communication partner (DCE/STE/IDTE).

6. 5. Functions

6. 5. 1. Facilities/utilities in call setup and clear

The X.25 recommendation defines User Facilities, which are included in the call setup and clear packets. A few facilities are strictly local to the DTE/DCE interface (e.g. window size), while most facilities carry end-to-end information during call setup or clearing. Examples of such "global" facilities are Throughput Class, Fast Select, and Called Line Address Modified Notification (CLAMN).

Facilities are either "subscribed to" or "per call" basis. "Subscribed to" means that the user has been allowed to use a certain facility, and this is noted in the NMC database. A few facilities are both "subscribed" and "per call" (Charging Information, CLAMN, Call Redirection Notification). This means for example that the DCE will only allow/generate these facilities if they are also subscribed to.

Besides User Facilities the X.75 recommendation operates with Network Utilities. According to the recommendation, User Facilities requiring STE or transit network action, must be mapped into Network Utilities.

This mapping is described below.

When a call enters RxPAX through a DCE, only User Facilities (i.e. no Network Utilities) are present in call setup packets. This is true whether the communicating partner is a DCE or an STE.

When a call request comes in from another network, i.e. from an STE, the RcPAX STE will convey the User Facilities transparently, and only look at the Network Utilities, if the call is for transit. If the call is not for transit, i.e. the called address is within RcPAX, the RcPAX STE will examine the utilities, and map those utilities which have a facility-counterpart and are not strictly local, onto a User Facility. Examples of such utilities are Throughput Class Indication, Fast Select, Packet Size Indication and Called Line Address Modified Notification.

If a call comes from inside RcPAX (i.e. from the LIC) and a call request is issued to another STE, the treatment again depends on whether the call originated in RcPAX (not transit) or outside RcPAX (transit). If the call originated in RcPAX, the packet coming to the RcPAX STE from inside RcPAX will not contain utilities. In this case the STE will examine the facilities, and map those facilities having a utility-counterpart, and are not strictly local, onto a Network Utility. If the call was for transit, the utilities are checked while the facilities are conveyed transparently.

When a call request packet enters the RcPAX STE from another STE, the RcPAX STE - hereafter just STE - will inspect calling and called address in the packet. If address translation is not used, the DCE/STE can decide whether the call is a transit-call or not, by inspecting the DNIC of the called address.

This is not the case when address translation is in effect: the DCE/STE must query the ATS for the physical address given the logical address from the packet. The ATS answers whether the called address can be reached via a DCE or an STE.

If the answer from the ATS tells the STE, that the called DTE is connected to RcPAX via another STE, the call is a transit call. Otherwise it is not.

In the former case the STE should just check the Network Utilities. In the latter case the STE should map utilities onto facilities and discard remaining utilities.

The DCE/STE must always scan the facilities/utilities before deciding what to do with the field "called address" in the call request packet. This is necessary, because there may be an RPOA-selection facility/utility present, and for the DCE there may further be an NSAP address present as the CCITT specified DTE facility "Called address extension".

The RPOA utility will not be further commented, as it will not be ready until the 1988 version of the STE is implemented. The 1984 STE will, however, convey an RPOA facility transparently in the call request packet.

If the DCE finds an RPOA facility in the call request packet, it must route to an RcPAX STE that knows the DNIC in the RPOA facility. This means that called address is not translated, but left unchanged in the packet.

In case the DCE finds an NSAP address, it will ask the ATS software component to translate the NSAP to a DTE address and insert this address as called address in the packet.

6. 5. 2. Buffer flow

It is the DCE/STE's job to supply buffers to both the HDLC (direction towards the DTE) and the LIC (direction towards the net). This is done on an even share basis. The DCE/STE creates during configuration a number of buffers to contain user data, and delivers the buffers to a software component called the PoolManager. When the DCE/STE at a later time needs a buffer, it sends a request to the PoolManager.

Two or more DCE/STE's on an RC5000 may share a PoolManager if they are configured for the same buffer size. In this way DCE/STE's may share each others resources.

Apart from the PoolManager, the DCE/STE has an Internal PoolHandler used to manage "small" buffers of fixed size. All places where a "small" buffer can be used, it will. Examples are: reset timeout, restart timeout, rr-packets, clear indication packets and packets used to increase credit.

The detailed description of the buffer configuration in the DCE/STE is presented in ref. 8.

To avoid an excessive use of buffers it is important that the DCE/STE has means to perform flow control on each logical channel. Towards the DTE flow control is achieved by means of the window mechanism used in X.25-3. Towards the network flow control is achieved by means of "credits" which the DCE/STE may issue to another DCE/STE. This is described in section 6.5.3.

One of the DCE/STE's major tasks is to ensure a "reasonable" service level for its client: the DTE, represented by the active logical channels. To achieve this, the resources offered by the DCE/STE are buffers and CPU time. When resources are scarce the DCE/STE must ensure that no logical channel is totally starved, i.e. the channel is still serviced, but the performance may be degraded.

Input may enter the DCE/STE from two directions: the HDLC and the LIC, and the DCE/STE will supply buffers to both sides on an even share part. If the HDLC is starved, it will send RNR's to the DTE, thereby effectively preventing the DTE from sending data. If such a situation exists during a certain time determined by a DCE/STE parameter, the DCE/STE will start resetting logical channels in order to get buffers.

If the LIC is short of buffers, it will cancel its resequence queue, and eventually the DG net will start dropping buffers.

Thus, in a "normal" situation the buffers will flow smoothly, and the logical channels will be serviced when required.

6. 5. 3. Flow control

The flow control mechanism for the DCE and the STE are identical, in this subsection only the DCE is mentioned but the description is also valid for the STE.

Both in the X.25/3 and in the LIC protocol there are flow control tools. One design consideration concerning the concatenation of the X.25/3 logical channel and the lic from the LIC component is to use these tools to provide the communicating DTE's with a reasonable flow control mechanism.

6. 5. 3. 1. X.25/3 flow control tools

On an X.25/3 logical channel, the transmission of datapackets is controlled separately for each direction and is based on authorizations from the receiver. A window is defined for each direction and negotiation of window sizes on a per call basis may be made with the flow control parameter negotiation. There are no rules about acknowledgement strategy, so every strategy is legal from sending an acknowledgement when a packet is received to sending an acknowledgement when a whole window is received. See the recommendation for a more in depth description.

6. 5. 3. 2. LIC flow control tools and restrictions

On a lic, the transmission of datapackets is controlled separately for each direction and is based on authorizations from the receiver. A lic window is implicitly defined for each direction and entirely controlled by the LIC users by exchanging credits. A LIC user is only allowed to send a packet to the other LIC user if it explicitly has received credit to do so. In contrary to the X.25/3 flow control the acknowledgement and the authorization is NOT coupled together. The acknowledgement is done entirely by the LIC module.

The LIC protocol provides 2 ways of sending credit. Either by issuing an increase credit primitive or by piggy backing credit in a data indication. One primitive can contain more than one credit.

The LIC will ensure, independent of the LIC user, that a datapacket is not forwarded to the other LIC user before credit is received. So the LIC maintains a queue of data packets waiting for credits from the other end.

The LIC has a system dependent maximum window (presently 16).

See LIC (chap. 5) for an in depth description.

6. 5. 3. 3. Design considerations

A number of different aspects are important for the coupling of the 2 set of flow control tools to give a good flow control strategy.

1) There is a limited amount of storage to be used for buffers, and in order to prevent one logical channel from using all the buffers there is a maximum number of buffers that can be used for data transmission on one logical channel.

   On the data direction from the DTE to the LIC, the DCE will not open its receiving X.25/3 window if no credit is available to forward the packets to the other end. This limits the maximum number of buffers used on this direction to basically the size of the X.25 receiving window.

   On the data direction from the LIC to the DTE, the DCE has a configurable variable, one for all logical channels, that determines the maximum number of buffers that can be used.

2) No buffers are dedicated to a specific logical channel. This means that if there is no traffic on a logical channel this channel will not use any resources. The DCE is therefore often configured with a number of logical channels based on an average traffic amount. This again means that the DCE can come into a lack of buffer situation as described in section 7.5.2.

3) A DTE can implement X.25/3 acknowledgements in different ways. So the DCE can not rely on any specific implementation and should as far as possible be independent of the X.25/3 acknowledgements. Therefore the sending of credit depends on X.25/2 acknowledgements.

4) Credits should be sent to the other end as soon as possible in order not to delay data transmission more than necessary. Therefore a credit can be sent to the other end whenever a packet is received from the LIC software component.

5) The number of internal credit packets should be minimized and piggy packing should be used as much as possible in order to minimize the total number of packets in the DG net. Therefore credits can be accumulated up to a limit before they are sent.

Notice that 4) and 5) are contradicting, and the limit in 5) is a configuration parameter which value is a balance between 4) and 5).

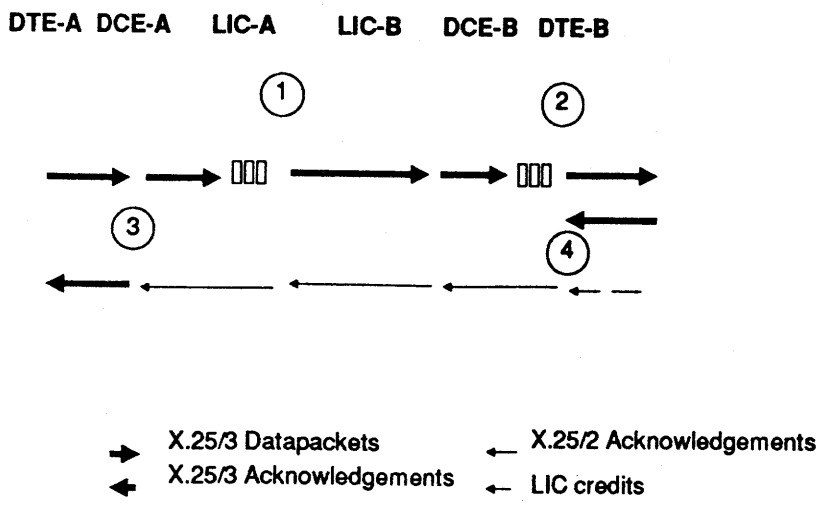
As a consequence of the above design considerations an account is kept as if a number of buffers was dedicated to support the data direction from the LIC to the DTE on one logical channel. Buffers are reserved when credits are sent to the other end and released, and thereby generating credit, when they are received from the X.25/2 software component. Decisions concerning credits are based on this accounting.

Explicit credit packets can be sent when either a data packet is received from the LIC or a buffer is returned from the X.25/2 software component. Furthermore the following conditions have to be met:

- Enough credits are accumulated.
- Buffers can be reserved.
- It is alloved to send credits according to the lic window.

When credits are piggy backed the first conditions are not relevant.

As described above there is an unavoidable interdependence between the X.25(X.75)/3 windows and the lic window. As both flow control tools work with authorization they will together give a back pressure mechanism which in a typical scenario will work as follows:



1) Based on available credit
2) Based on X.25/3 acknowledgements
3) Based on available credit
4) Credits may be accumulated

Figure 71 Back Pressure Mechanism

1) The DTE-B does not open its receiving window - I.e. DCE-B does not receive X.25/3 acknowledgements.

2) Packets arrive from the LIC-B, DCE-B sends credits to DCE-A until there are so many packets waiting in the DCE-B that there are no more free resources.

3) The LIC-A sends packet until all credit is used.

4) The DCE-A forwards all packets received, but does not sent any X.25/3 acknowledgements when there is no more credit.

5) The DTE-A will close the X.25/3 window and then the data transmission in this direction will stop.

6. 5. 4. The M-bit concatenation mechanism

The ratio (control data/user data) gives a theoretical limit for the line utilization of the access line. The smaller ratio the better line utilization. If the node is configured so there is only one trunk line for one access line it is the maximum ratio on the access line and the trunk line that gives the theoretical limit. The ratio on the trunk line is the larger. In such a configuration the ratio on the trunk line will therefore influence the possible line utilization of the access line.

In order to minimize the ratio on the trunk line the DCE/STE can concatenate 2 X.25/3 packets and send them as one datagram on the lic. By using the X.25/3 M bit the DTE determines on a packet basis when concatenation is wanted.

When the DCE/STE is configured for it and when an X.25(X.75)/3 category A packet (full packet with mbit = 1 and dbit = 0) is received from the DTE(STE) the packet will be concatenated with the following packet independently of the category. The concatenated packet will be sent through the datagram network with the category of the second packet. Concatenation of packets is only done when the size of the data field of the concatenated packet does not exceed the internal data field size (presently 256 bytes).

When the DCE/STE receives a packet from the LIC with a user data field larger than agreed for this logical channel it must be a concatenated packet. The packet is then unpacked in a category A packet and in a packet with the same category as the concatenated packet.

Because the concatenated packet is sent through the datagram network as one packet it is credit wise treated as one packet. The DCE/STE compensates for this in the credit accounting.

7. The Internal DTE (IDTE)

To appreciate the name "Internal DTE" consider a network layer service of a DTE attached to an RcPAX network, see fig. 72.

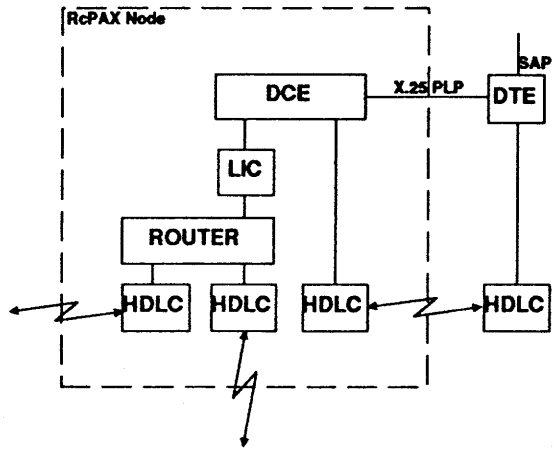


Figure 72 DTE attached to RcPAX

An Internal DTE is an entity that resides in a node internal to an RcPAX network and provides a DTE's network layer service, (referred to as internal access service), to a user entity residing in the same internal RcPAX node, see fig. 73 .

This "IDTE approach" to an internal access service saves transmission equipment, transmission delay and memory as the X.25 protocol part can be cancelled out and allows the management of the DTE part of the IDTE to be directly integrated with RcPAX's network management system via the LCP service.

Despite the fact that an IDTE entity resides in RcPAX' network layer and provides a "DTE network layer service" it does not constitute a complete OSI network service access point of an OSI end system, see fig. 74 .

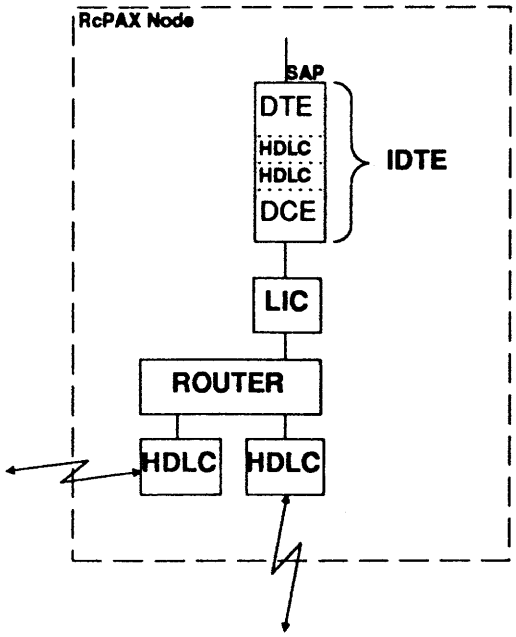


Figure 73 Conceptual structure of the IDTE

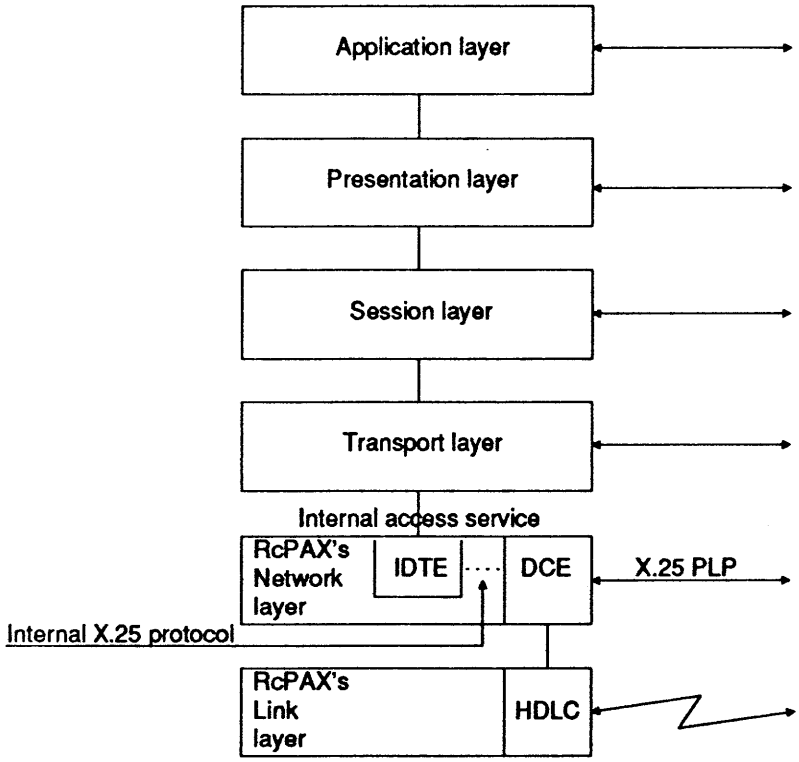


Figure 74 RcPAX internal access service in OSI perspective

The RcPAX internal access service is similar to the OSI Network service but not as abstract. Rather its abstraction level corresponds to the Network Internal Layer Service (NILS) defined in ref. 10 and 11) as packetizing of NSDUs into M-bit sequences, NSAP routing and some aspects of Quality Of Service is not a part of the internal access service and must be performed by IDTE users. Furthermore it is, at least, unconventional to use a genuine CCITT X.25 DCE role in an OSI-endsystem for the Network Access protocol.

The internal access service is used by the software components that provides the associated services, cf. section 3.4, and in general software components that provides value added network services. Eg. Internal PAD- and Test Answer Back from Network software components make use of the internal access service.

To each IDTE user entity corresponds exactly one IDTE entity, which in turn is connected to exactly one LIC entity. It is therefore seen that an IDTE entity is engaged in two interfaces: on one side towards a LIC entity and on the other side towards a IDTE user entity, see fig. 75 .

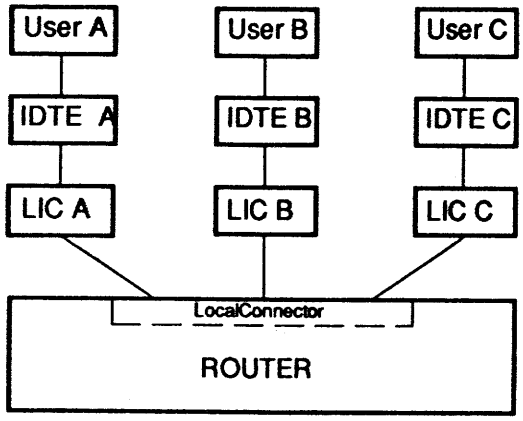


Figure 75 Entity Interfaces

The IDTE user/IDTE interface (also just called the IDTE interface) is an internal software interface within the node. The IDTE interface of a single IDTE entity contains a single internal service access point.

Due to the one-to-one relationship between an IDTE entity and its LIC entity, and an IDTE entity and its IDTE user entity, this in fact implies that also all IDTE user entities are uniquely addressable within the RcPAX datagram address space. That is, the set of addresses of access points of this internal access service can be mapped into the set of datagram addresses of an RcPAX network.

There exists a "normal" DTE software component (cf. fig.), referred to as an external DTE (XDTE) with a service interface having a large subset in common with a large subset of the IDTE interface. The main differences are that the XDTE interface does not support the datagram pass through service, on the other hand it supports several service access points - i.e. allows more than one user. In fact there exists users (eg. PAD and TABN) that do not need to know whether it is serviced by an IDTE or an XDTE. To achieve this, some elements of the IDTE interface has been introduced even if they are somewhat artificial. (The optional window size negotiation facility is an example of this).

As described in section 3.3 about access services IDTE entities communicates in "peer-to-peer" relationships with DCE-, STE- or IDTE entities, where DCE- and STE entities are relays to external access service rather than providers of internal access service. Anyhow, to ease the following description, a DCE- or STE entity will in some circumstances be considered as structured as an IDTE- and an IDTE- user entity, see fig. 76.

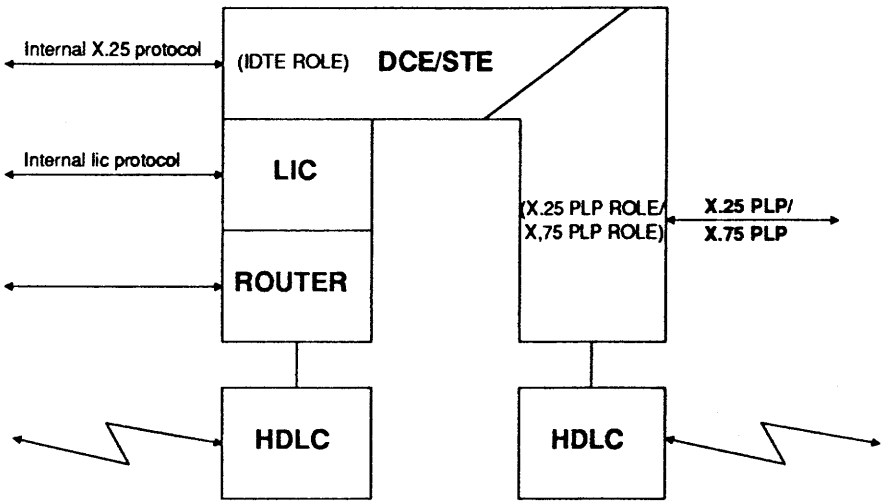


Figure 76 DCE/STE entity with IDTE and IDTE-user roles

The part of the DCE-/STE entity that performs the IDTE role is considered to be an IDTE entity, and the part that performs the X.25 PLP role is considered to be an IDTE-user entity. In the following, this view will be applied, when reference is made to the remote "peer-to-peer" communication partner of an IDTE. That is, the communication partner may actually be a DCE or STE entity even if it is referred to as an IDTE.

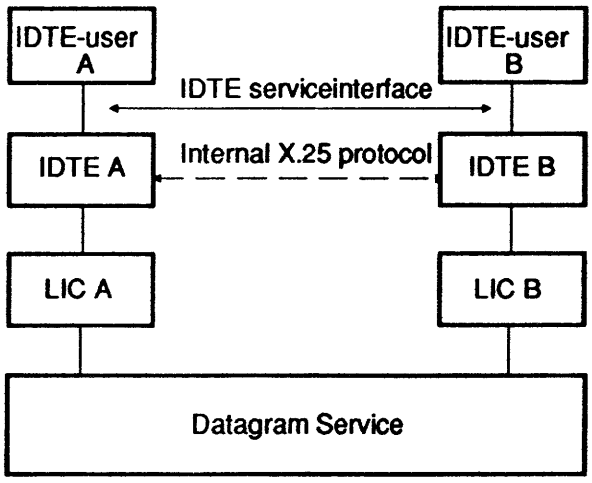


Figure 77 The IDTE service

7. 1. The IDTE Services

The IDTE services are provided on the IDTE interfaces of two cooperating IDTE entities running the internal X.25 protocol on top of the LIC service. The IDTE service is in turn made use of by the two corresponding IDTE user entities, see fig. 77.

The end-points of connections on which the IDTE services are provided on the IDTE interface are termed streams. An IDTE entity can handle a number of streams simultaneously, see fig. 78 . The stream identifier is provided by the IDTE entity. The IDTE ensures that streams are unambigously identifies within the scope of the IDTE interface in question. Each stream identifier has a unique alias called on logical channel identifier. The mapping between stream identifiers and logical channel identifiers is determined by configuration. Logical channel identifiers are used to identify streams in relation to Network Management and the IDTE users utilization of PVCs.

A stream is of one of three kinds - SVC, PVC or Datagram - corresponding to the service it provides. That is , the concepts of switched virtual calls, permanent virtual circuits and datagram channels (cf. ref. 8) are mirrored rather explicitly at the IDTE interface.

The stream identifiers together with the unique RcPAX datagram address of the IDTE entity, make the streams uniquely identifiable within an RcPAX network. It is seen that all streams on an IDTE interface have a common datagram address which is that of the IDTE entity supporting the streams, see fig. 78 .

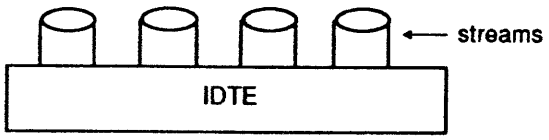


Figure 78 The IDTE ports (streams)

An association between a pair of two (non Datagram) streams is denoted an internal virtual call (ivc). Associations between Datagram streams exist only implicitly at this level. Only a single ivc can be handled between a pair of

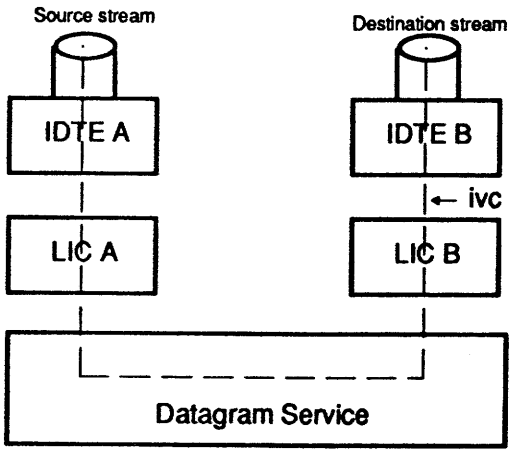


Figure 79 The internal virtual call (ivc)

streams. The ivc is identified by the combination of the identification of the two involved streams, called the source stream and the destination stream respectively, see fig. 79 .

The following elements of the IDTE service are considered:

- the addressing and routing service
- the supervisory service
- the packet transfer service
- the interrupt transfer service
- the datagram pass through service
- the flow control service
- the reset service
- the receipt confirmation service
- the optional IDTE-user facilities

(Only the addressing and the datagram pass through service is relevant for Datagram streams).

The addressing and routing service
The internal access service allows address information to be conveyed unchanged during ivc establishment. (If no calling address information is present then the IDTE inserts its own configured so called DTE address).

Called address information is used by the IDTE to determine an RcPAX network address. The IDTE considers the called address information to consist of two parts, see fig. 80.

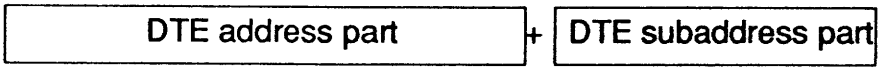


Figure 80 Address information parts

The IDTE interprets only the DTE address part. The interpretation of the DTE subaddress part is entirely up to the users of the access service.

The DTE address part assumed to have CCITT X.121 format. I.e. the DTE address part is considered as consisting of two parts, see fig. 81.

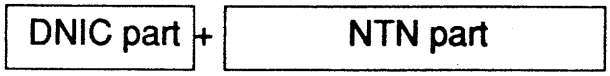


Figure 81 DTE address parts

The IDTE routes the following way:

- If the DNIC part is equal to the IDTE's configured own DNIC, then the NTN part is directly used as the RcPAX network address.

- If the DNIC part is not equal to the IDTE's configured own DNIC then all RcPAX network addresses, that the address of the DNIC part is bound to, is looked up by means of the DIS, and if more than one address is retrieved an arbitrary address is selected as the RcPAX network address.

(The IDTE interface does not support general address translation of address information like DCE and STE. That is, OSI NSAP addressing is not supported in general).

The supervisory service
The supervisory service is used to control ivcs. It includes services to establish, maintain and remove ivcs.

The packet transfer service
On an established ivc, the packet service offers means of transfer of packets between the two involved streams. The service is full duplex meaning that the flow of packets in the two directions are handled independently of each other. The packet service provides sequenced delivery of the packets.

The IDTE does not support fragmentation and assemblation of packets in general. But the IDTE service is equipped with means to negotiate with the remote IDTE, for each direction independently, a common maximum packetsize of packets of an ivc. An "m-bit" may be set in each packet, but it is transferred transparently - leaving its utilization entirely up to the IDE-user.

Single packets may be segmented and assembled by lower layer services - see the lic service description.

The interrupt transfer service
The interrupt service is also offered on an existing ivc. Both the packet and the interrupt service are offered simultaneously on the streams.

The interrupt service offers delivery guarantee but no flow control - except that the idte ensures that, on a single ivc, at most one interrupt packet is outstanding at a time. Like the packet service, the interrupt service is full duplex, i.e. interrupt packets are transferred independently in the two directions on a stream.

As for the packet service the IDTE-user entities are informed through the removal of the stream if the guarantees of the interrupt service cannot be lived up to.

The Datagram pass through service
This service allows IDTE-users to exchange datagrams between them using the IDTE interface. No association between the IDTE user entities shall be established first. (The use of stream identifiers in invocations of the datagram service does not imply that any ivc actually exists. They corrrespond to the Datagram channels, cf. ref. 8).

The IDTE entities map this service directly onto the datagram service provided by the LIC software component, so no delivery, sequencing and non-duplication of datagrams are guaranteed.

The flow control service
For the packet service a mechanism enabling the IDTE entities to control the flow of packets on an ivc is offered.

This flow control mechanism is based on a simple credit scheme: An IDTE entity is not allowed to send a packet before it has obtained a credit from the other (potentially receiving) IDTE entity. A credit can be seen as a token allowing one ivc-packet to be sent. Each IDTE entity issues a number of credits according to its current receiving capabilities. The issued credits are transported to the other IDTE entity by means of the LIC service, telling it how many packets it may send. For each packet the IDTE entity sends on an ivc, the number of available credits is decreased by one.

The rate of which an IDTE issues new credits depends on the receive mode of the stream, see the paragraph on input buffers in section 7.2. If the receive mode of the stream is "dedicated" then no credit is issued initially when the ivc is established or reset, and a credit is only issued when a receive buffer for the stream is delivered by the IDTE-user entity. If the receive mode of the stream is "general" then a configured amount of credit is issued initially when the ivc is established or reset, and a credit is issued as soon as an indication of data is received from the remote IDTE entity.

That a credit is issued does not necessarily mean that it will be sent immediately. Issued credit may be accumulated internally in the IDTE entity in order to piggy back it. The IDTE interface features means for the user to determine a threshold for the local accumulation of issued credit in the IDTE entity of the remote end of an ivc during ivc establishment. (The optimal threshold is determined from knowledge about the traffic pattern, e.g. one-way or interactive, the transit delay and the number of output buffers normally at the IDTE). (Actually this threshold can only be controlled if the remote communication partner is an IDTE entity and not a DCE/STE entity. Their thresholds are determined by local means).

If this threshold is exceeded then credits are sent by means of the LIC service.

The transfer of credits between IDTE entities is independent of the packet transfer, i.e. even if packet transfer is blocked in one direction, credits for packet transfer in the opposite direction can be issued/delivered.

The Reset service

In some (failure) situations an IDTE-user entity may require an established ivc to be reset. A reset forces an ivc and the transfer of packets and interrupts into a well-defined state, from which the communication reliably can be resumed. During reset are pending packets and messages (i.e. in transfer between the two IDTE-user entities) are dropped and the delivery control are suspended. This implies that the IDTE-user entities have no knowledge of whether or not packets/interrupts sent and being in transfer at the initiation of a reset actually are delivered to the other IDTE-user entity.

However, the packet and interrupt services guarantee that packets/interrupts sent after a reset will not be delivered to the other IDTE-user entity before that IDTE-user entity has been informed of the reset. A reset can therefore be seen to form a synchronization point in the communication between the two IDTE-user entities.

Except in relation to PVC streams, is a reset always initiated by one (or both) of the IDTE-user entities, never by the IDTE entities.

All credits are in principle withdrawn during reset, but in some cases, e.g. if the receive mode of the stream is general, a number of credits is sent during the reset.

The receipt confirmation service

The IDTE interface offers means to request for confirmation of receipt. On the other hand, the interface does not provide a packet- acknowledge primitive. If an IDTE entity receives a packet from the remote IDTE entity with the confirmation request parameter set it itself issues an acknowledge packet when it has indicated the received packet to its users.

The optional user facilities

The following optional user facilities are defined in relation to the internal access service.

Subscription based (pertinent to all streams):

General (cf. ref. 6):

Facility	Comment
Incoming calls barred	
Outgoing calls barred	
Non-standard default packet size	both directions
Non-standard default window size	both directions
Default throughput class assignment	both directions
Flow control parameter negotiation	
Throughput class negotiation	
Closed user group	
Closed user group with outgoing access	
Closed user group with incoming access	
Incoming calls barred within a closed user group	
Outgoing calls barred within a closed user group	

Fast select authorization	currently dummy
Fast select acceptance	currently dummy
Reverse charging authorization	
Reverse charging acceptance	
Charging information authorization	always on
NUI subscription	NUI selection not mandatory in every call

IDTE specific:

Facility	Comment
Dynamic tariff assignment	
Remote NUI selection indication	only "TA-part" is indicated
Default initial issued credit assignment	
Receipt confirmation authorization	always on
Receipt confirmation negotiation	always on
More data bit authorization	always on
Qualifier bit authorization	always on

IDTE interface based (per stream):

General (cf. ref. 4):

Facility	Comment
Closed user group selection, basic format	
Fast select	currently dummy
Reverse charging	
Charging information request	
Charging information, segment count indication	
Charging information, call duration indication	
Flow control parameter negotiation, window size	
Throughput class negotiation	no inpact on throughput
NUI selection	
Calling address extension	
Called address extension	

IDTE specific

Facility	Comment
DNIC routing selection	
Receive mode assignment	
Remote credit threshold assignment	

Receipt confirmation request
More data information
Qualifier information

Priority

7. 2. The IDTE service interface

This section describes the IDTE interface through which the IDTE services as outlined in the preceeding section are provided.

The following description based on service primitives provides an overview of the flow of information between the involved IDTE entities (the service-provider) and the LIC user entities (the service-users) without going into implementation details.

Before the service primitives are described, some comments on the implementation of the IDTE service interface are provided as follows.

The implementation of the IDTE interface is based on the mechanisms of software processes communicating by exchanging buffers via buffer queues.

This means that the IDTE interface "by nature" is asynchronous implying that the IDTE entity and the IDTE-user entity may have different perceptions on the state of their interface at a given point in time. Therefore collisions may happen on the IDTE interface, which have to be resolved. This is especially the case when resetting an ivc.

All buffers on the LIC interface are provided by the IDTE-user entity. The buffers are said to be "signalled" from the IDTE-user entity when handed over to the IDTE entity and "returned" in the opposite direction.

In general two types of buffers are considered on the LIC interface:

Output buffers
These buffers are used to carry request and response primitives from the IDTE-user entity to the IDTE entity. An output buffer is pending at the IDTE entity as long as the execution of the request/response primitive proceeds and then returned to the IDTE-user entity, i.e. the buffers may be pending at the IDTE entity for a rather long time. The IDTE entity puts some information indicating the result of the execution of the request/response in the returned buffer (and the reason in case of failure). This information can normally be ignored by the IDTE-user entity.

The output buffers are used by the IDTE entity to send internal X.25 protocol data units to the remote IDTE entity, i.e. the buffers may be signalled along to the LIC entity and be pending there while being handled, (that is, they may further be signalled along to the ROUTER entity).

Input buffers
Input buffers are used to carry indication and confirm primitives from the IDTE entity to the IDTE-user entity. An input buffer is signalled by the IDTE-user entity to the IDTE entity and will be pending until it is returned to the IDTE-user entity carrying an indication or confirm primitive.

Two kinds of input buffers exist, general and dedicated input buffers. The IDTE entity signals the general input buffers along to the LIC entity as a normal input buffer in order to receive internal X.25 protocol data units sent by the remote IDTE entity. As the LIC entity forwards the input buffer to the ROUTER entity it is pending there until a protocol data unit is received (an RcPAX datagram) and then returned to the IDTE entity. If the received protocol data unit is a data packet on a stream in "dedicated" receive mode the data will be copied into a dedicated receive buffer (guaranteed available by the credit mechanism to be pending) which then is returned to the user indicating a data packet, and the general receive buffer will be signalled back to the LIC entity. If received protocol data unit is not a data packet or is a datapacket on a stream in "general mode" the IDTE entity may return the buffer issuing an indication or confirm primitive to the IDTE-user entity.

It is the responsibility of the IDTE-user entity to provide the required amount of general input buffers. If it fails to do so, the ROUTER entity cannot deliver the datagrams to the LIC and IDTE entities at the required rate. The ROUTER entity will then start dropping datagrams, which will trigger a number of recovery mechanism in the "lic service layer". The overall performance will be degraded significantly.

Dedicated receive buffers are withheld by the IDTE entity (not passed along to the LIC entity). Each of these buffers are pertinent to a specific stream in "dedicated" mode, and as indicated above they are used to indicate to the IDTE-user received data packets on these streams.

Thus, general receive buffers are used to return X.25 control packets to the IDTE users, as received buffers shared between the streams in "general" mode and as the transportation mechanism from ROUTER via LIC to IDTE of received data on dedicated streams.

7. 2. 1. Ivc Establishment

The establishment of an ivc is performed using the ESTABLISH primitive:

- the calling IDTE-user entity issues an ESTABLISH request,
- the results in an ESTABLISH indication to the called IDTE-user entity,

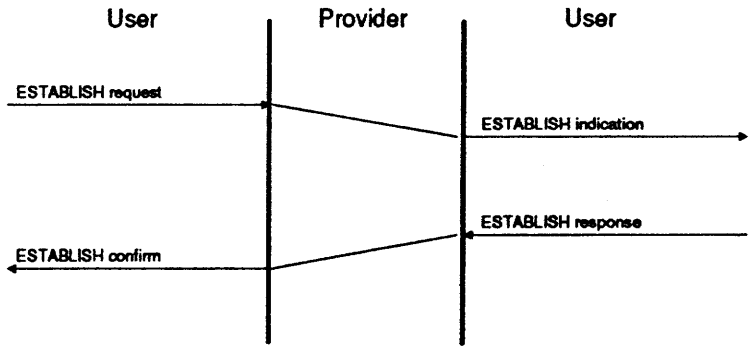


Figure 82 Ivc establishment

- this results in an ESTABLISH indication to the called IDTE-user entity,ndependent in eachthe called IDTE-user answers the ESTABLISH indication by issuing an ESTABLISH response,

- the ESTABLISH response results in an ESTABLISH confirm to the calling IDTE-user entity.

See fig. 82 .

The IDTE-user entities may start using the ivc for packet and message transfer as soon as the ivc is completely established for each of the IDTE-user entities, that is:

- when the calling IDTE-user entity receives the ESTABLISH confirm,

- when the called IDTE-user entity issues the ESTABLISH response.

The ivc establishment may fail for many reasons. In all phases of the ivc establishment process both the IDTE and IDTE-user entities may initiate the removal of the partly established ivc, see section 7.2.2.

An important case is when the called IDTE-user entity for some reason does not establish the new ivc, e.g. no more resources. In that case, the called IDTE has to initiate the removal of the ivc.

No collision resolution exists in relation to ivc establishment, so if the two IDTE-user entities simultaneously initiate ivc establishment to each other, two ivcs will (attempted to) be established.

7. 2. 2. Ivc removal

The removal of an ivc can be initiated either by the IDTE-user entities or by the IDTE entities.

For IDTE-user initiated removal:

- the removing IDTE-user entity initiates the removal by issuing a REMOVE request,

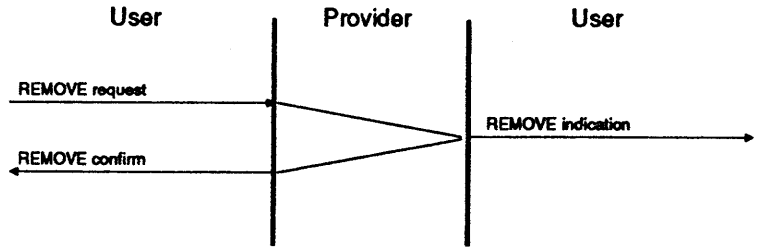


Figure 83 Ivc removal (IDTE-user initiated)

- if the ivc is at least partly established at the other IDTE-user entity, i.e. it has received an ESTABLISH indication (and no removal has yet been initiated), a REMOVE indication is issued to it,

- when the ivc has been removed, a REMOVE confirm is issued to the removing IDTE-user entity.

See fig. 83 .

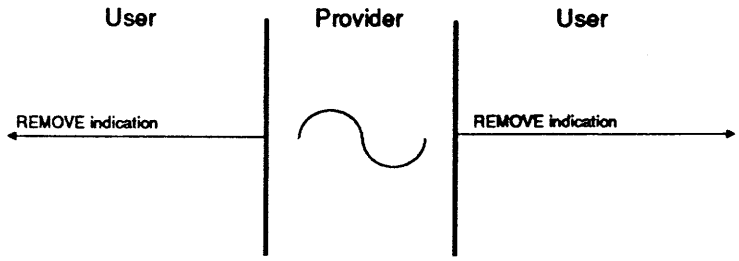


Figure 84 Ivc removal (IDTE initiated)

For IDTE initiated removal:

- the removing IDTE entity issues a REMOVE indication to its local IDTE-user entity,

- if the ivc is at least partly established to the remote IDTE-user entity, i.e. it has received an ESTABLISH indication (and no removal has yet been initiated), a REMOVE indication is issued to it.

See fig. 84 .

An IDTE-user entity shall consider the ivc to be removed when it has received either a REMOVE confirm or REMOVE indication, i.e. it must not issue any primitive on this ivc anymore.

Packets and interrupts in transfer on an ivc may be lost during the remove, i.e. no delivery guarantee.

As all the involved entities are allowed to initiate the removal of the ivc, a number of collision situations may occur. There are resolved by the IDTE entities and the rule that

- an IDTE-user entity that has issued a REMOVE request shall regard a received REMOVE indication as a REMOVE confirm.

In fact, except for some information passed in the primitives, the implementation of the IDTE interface does not distinguish between the REMOVE confirm and indication primitives.

The IDTE initiated removal procedure takes precedence of the IDTE- user initiate procedure, so this may at any point be interrupted by the IDTE initiated procedure.

If the IDTE entities encounter any problems executing the IDTE-user initiated remove procedure, the IDTE initiated procedure is started.

7. 2. 3. Ivc Reset

The IDTE-user entities may initiate a reset of an established ivc, i.e. the

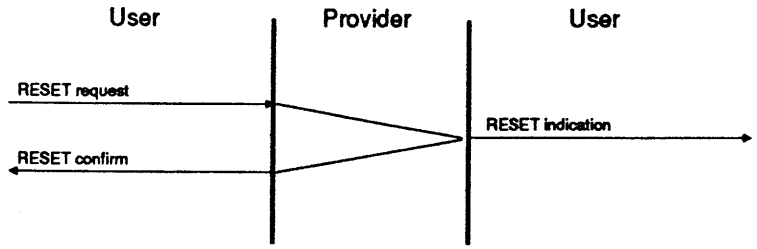


Figure 85 Ivc reset

establish procedure must have been completed and no remove procedure initiated:

- the resetting IDTE-user entity issues a RESET request,
- this results in a RESET indication issued to the remote IDTE-user entity,
- when the reset of the ivc is finished, a RESET confirm is issued to the resetting IDTE-user entity.

See fig. 85.

The IDTE-user entity shall consider the ivc to be reset when it receives the RESET indication or RESET confirm as appropriate.

The IDTE-user entities do not know the delivery status of packets and messages sent before the reset was initiated, but after the reset the IDTE-user entities receive only what the other IDTE-user entity has sent after the reset.

All previously issued credits are implicitly withdrawn by the reset, however, the IDTE issues credit in the RESET request and confirm request corresponding to the initial credit if the stream is in "general" mode or to the current

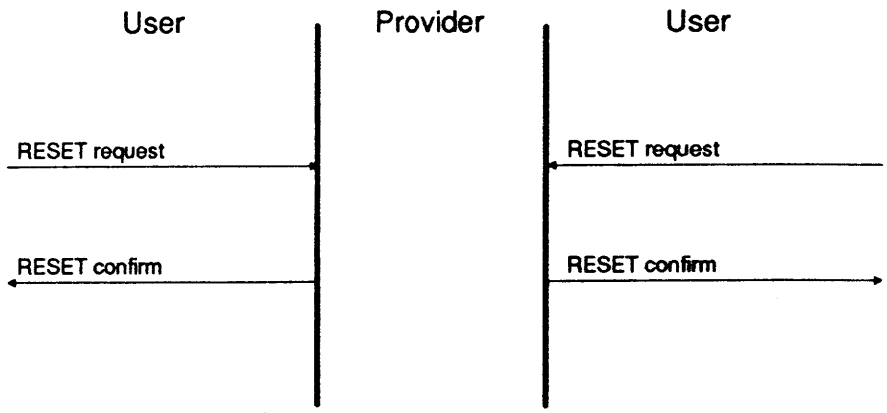


Figure 86 Ivc reset collision

available number of dedicated receive buffers if the stream is in "dedicated" mode. (Note that it is configurable whether the dedicated buffers of a "dedicated" stream is returned to the user entity or not during the reset).

Both of the ivc remove procedures take precedence of the ivc reset procedure, so all of the involved entities may abort the reset procedure by starting a remove procedure, see 7.2.2.

Both the IDTE-user entities may invoke the reset procedure simultaneously by issuing RESET requests. This collision is resolved by the IDTE entities so both IDTE-user entities will receive RESET confirm and no RESET indication as illustrated in fig. 86.

A reset collision may also take place between an IDTE-user entity and the IDTE entity, i.e. the IDTE-user receives a RESTART indication while waiting for a RESET confirm. In this case the ivc is reset twice, first reset corresponding to the RESET indication and then reset corresponding to the RESET request and related RESET confirm.

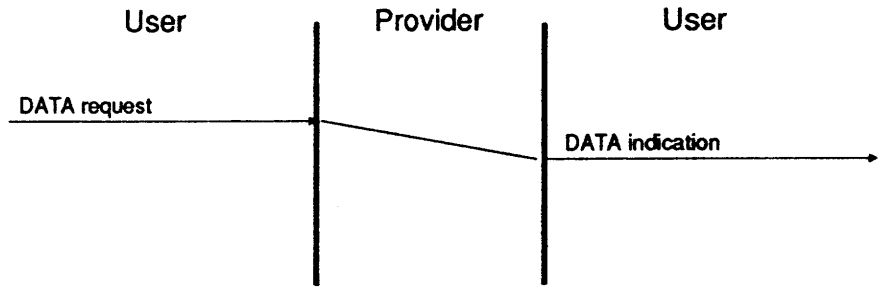


Figure 87 Packet transfer

7. 2. 4. Packet Transfer

The DATA primitives are used for transfer of packets between the two involved IDTE-user entities:

- the sending IDTE-user entity issues a DATA request,
- this results in a DATA indication issued to the receiving IDTE-user entity.

See fig. 46 .

If possible, the IDTE-user entities will normally issue a number of DATA requests to "smooth" flow of packets. DATA indications are issued in the same sequence as the corresponding DATA request. DATA requests which cannot be handled for the time being by the IDTE entity to which they are issued, will be queued as appropriate.

The remove and reset procedures take precedence of the packet transfer procedure. Previously issued DATA request may be lost during the remove or reset procedures.

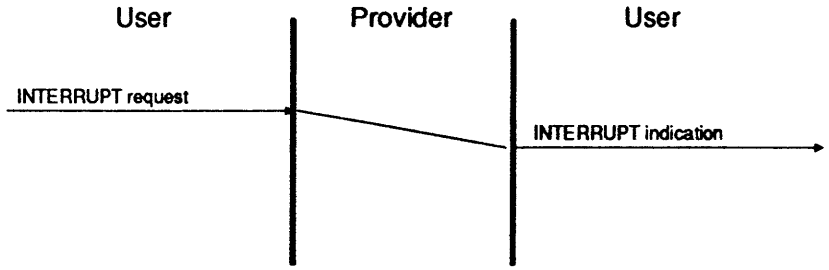


Figure 88 Interrupts transfer

7. 2. 5. Interrupt Transfer

Interrupts are handled on an ivc using the INTERRUPT primitives:

- the sending IDTE-user entity issues an INTERRUPT request,
- this results in an INTERRUPT indication issued to the receiving IDTE-user entity.

See fig. 88 .

An IDTE entity will only execute one INTERRUPT request at a time.

If an IDTE-user entity issues more than one, the INTERRUPT requests not being executed are queued inside the IDTE entity receiving the INTERRUPT request.

The remove and reset procedures take precedence at the interrupt procedure, i.e. it may be interrupted at any time. Interrupts may be lost during the remove and reset procedures.

7. 2. 6. Datagram Transfer

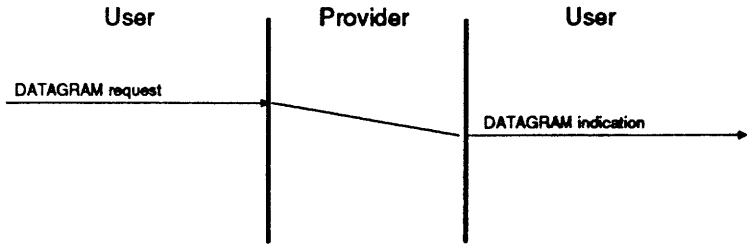


Figure 89 Datagram transfer

An IDTE-user entity may request the IDTE entity to transfer a datagram using the DATAGRAM primitives:

- The sending IDTE-user entity issues a DATAGRAM request.
- This results in a DATAGRAM indication issued to the receiving IDTE-user entity.

See fig. 89 .

The DATAGRAM indication may or may not be generated; there is no delivery guarantee.

This procedure is not influenced by any of the other IDTE service procedures.

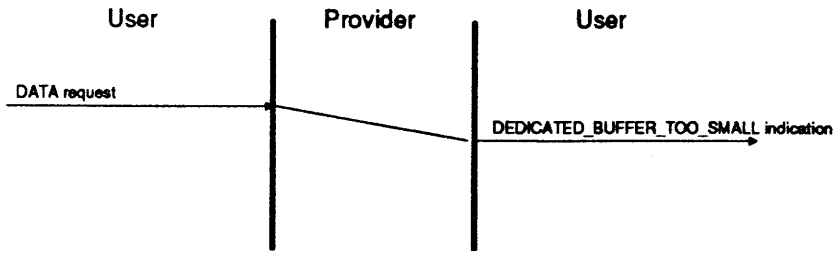


Figure 90 Dedicated buffer too small

7. 2. 7. Dedicated receive buffer too small

It is up to the IDTE-user entities to ensure that the dedicated receive buffers can hold the packets received from the remote IDTE- user. (The IDTE interface supports end-to-end packetsize negotiation during ivc establishment).

If the IDTE entity receives a packet from the network and the corresponding dedicated receive buffer is too small to hold this packet then the IDTE entity will indicate this to the IDTE-user. The packet is not even partly indicated to the IDTE-user entity but is discarded by the IDTE entity (see fig. 90). It is up to the IDTE-user entities to recover from this situation.

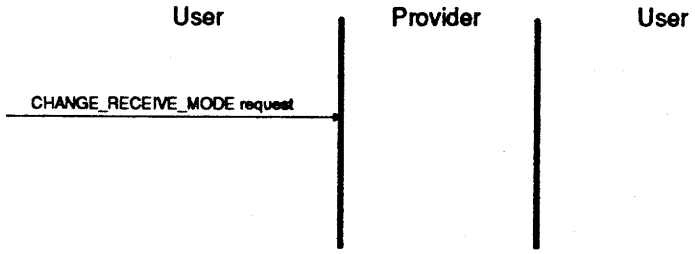


Figure 91 Change receive mode

7. 2. 8. Change Receive Mode

The receive mode of a stream is determined during ivc establishment, but if the receive mode initially is "dedicated" the IDTE-user entity may dynamically change the receive mode to be "general":

- The IDTE-user entity issues a CHANGE_RECEIVE_MODE request.

See fig. 91 .

8. References

1. RcPAX
An Introduction
Edition 2.0 Denmark
June 1990

2. RcPAX Network Management System
An Introduction
1990

3. RcPAX Load System
General Description
PN: 99001398

4. RcPAX Operating System
General Description
PN: 99001418

5. RcPAX Name Server
General Description
PN: 99001433

6. CCITT
Red Book 1984
Volume VIII- Fascicle VIII.3
Data Communication Networks
Recommendation X.25

7. CCITT
Red Book 1984
Volume VIII - Fascicle VIII.4
Data Communication Networks
Recommendation X.75

8. CCITT Recommendation X.25
Yellow Book 1980
Volume VIII - Fascicle VIII.2

9. RcPAX Network System
X.25 DCE and X.75 STE
Installation and Operating Guide
PN: 99001413

10. ISO/IEC DIS 10028
Definition of the relaying functions of
a network layer intermediate system -
Part 2: Connection-mode network service
April 1991

11. ISO/IEC DIS 10177
Intermediate system support of the OSI
connection-mode network service using
ISO/IEC 8208 in accordance with ISO/IEC 100628
June 1990

PN: 99001391