ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-92

CONNECTIONLESS INTERNETWORK PROTOCOL

March 1984

Free copies of this document are available from ECMA,
European Computer Manufacturers Association
114 Rue du Rhône – 1204 Geneva (Switzerland)

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-92

CONNECTIONLESS INTERNETWORK PROTOCOL

March 1984

BRIEF HISTORY

This Standard ECMA-92 is one of a series of Standards for Open Systems Interconnection.

Open Systems Interconnection Standards are intended to facilitate homogeneous interconnection of heterogeneous information processing systems.

The Standard is within the framework of the co-ordination of Standards for Open Systems Interconnection which is defined by ISO/7498. It is based on the practical experience of ECMA member companies worldwide, and the results of their active participation in the current work of ISO, the CCITT, IEEE and national standards bodies in Europe and the USA. It represents a pragmatic and widely based consensus.

Adopted by the General Assembly of ECMA as Standard ECMA-92 on December 13,1983

TABLE OF CONTENTS

1 General

1. GENERAL

1.1 Introduction

The protocol defined in this Standard ECMA-92 is intended as a common subnetwork-independent means to communicate across separate heterogeneous subnetworks, in order to provide a larger combined network with a homogeneous connectionless service.

Though a connectionless global network service is still under study within ISO, ECMA identified the need for the standardization of a Connectionless Internetwork protocol as it is a pragmatic approach to satisfy urgent needs for LAN interconnection. For this approach the following assumptions have been made:

- If several LAN are interconnected no subnetwork enhancement function in sublayer 3b need to be performed.

- If an OSI end system consists of several interconnected (possibly via WAN) LAN, it is assumed that the quality of transport service maintained by this end system is high compared with the quality of service provided by the network connection to and from this end system.

It is the intention of ECMA to continue its work with the definition of a Connectionless Internetwork protocol as a separate Standard.

The Internetwork protocol operates between the peer entities of sublayer 3c of the Network Layer (reference ECMA TR/13 and TR/14). It is closely related to the Network service and is built upon the services provided by subnetworks. The interrelationship of this Standard with these services is depicted in Figure 1.

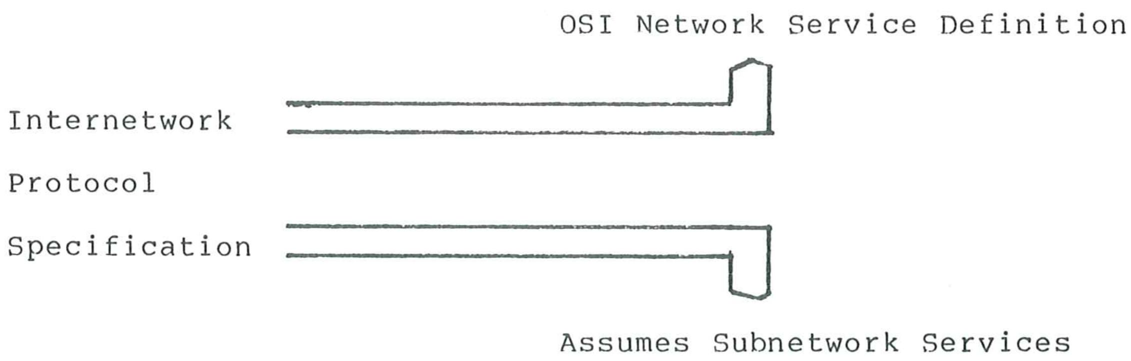


Fig. 1 Interrelationship with Adjacent Services

Some examples of possible applications of the protocol are:

- the interconnection of a number of LAN on a single site;
- the interconnection of LAN via a public network.

This Standard defines the Connectionless Internetwork protocol using both the English language and a formal description. The formal description is intended to clarify and make explicit the English language description where ambiguity might otherwise arise.

1.2 Scope

This Standard ECMA-92 specifies:

- procedures for the connectionless transmission of data and control information from one internetwork entity to a peer internetwork entity;

- the encoding of the internetwork protocol data units used for the transmission of data and control information;

- procedures for the correct interpretation of internetwork protocol control information;

- the functional requirements for implementations claiming conformance to the Standard.

The procedures are defined in terms of:

- the interactions among peer internetwork entities through the exchange of internetwork protocol data units;

- the interactions between an internetwork entity and a network service user through the exchange of Network Service Primitives; and

- the interactions between an internetwork entity and a subnetwork dependent service provider through the exchange of subnetwork dependent service primitives.

This Standard ECMA-92 specifies a Connectionless Internetwork protocol. This protocol relies upon the provision of a connectionless subnetwork service.

A specific need has been identified for a subset of the Connectionless Internetwork protocol appropriate to simple interconnection of Local Area Networks, either directly or via a single Wide Area Network. This Standard includes the specification of such a subset as a simplified form of the protocol.

1.3 References

ISO/7498	Data Processing - Open Systems Interconnection - Basic Reference Model
ECMA-72	Transport Protocol (June 1982)
ECMA TR/13	Network Layer Principles
ECMA TR/14	Local Area Networks - Layer 1 to 4 Architecture and Protocols
ECMA TR/20	Layers 4 to 1 addressing
ISO/97/16 N1347	A Formal Description Technique based on an Extended State Transition Model
ISO/97/6 N2613	Internal Organization of the Network Layer
ISO/97/6 N2713	Information Processing Systems - Data Communications - Addendum to the Network Service Definition Covering Connectionless Data Transmission

1.4 Definitions

1.4.1 Reference Model Definitions

The following terms in this Standard ECMA-92 have the definition given in ISO 7498:

1.4.1.1 Network-Entity

An active element within a Network-Subsystem.

1.4.1.2 Network Layer

A subdivision of the OSI architecture, constituted by subsystems of the same rank.

1.4.1.3 Network Protocol

A set of rules and formats (semantic and syntactic) which determines the communication behaviour of Network entities in the performance of Network functions.

1.4.1.4 Network Protocol Data Unit

A unit of data specified in a network-protocol and consisting of network-protocol- control-information and possibly network-user-data.

1.4.1.5 Network Relay

A network-function by means of which a network entity forwards data received from one correspondent network-entity to another correspondent network-entity.

1.4.1.6 Network Service

A capability of the Network layer and the layers beneath it, which is provided to transport entities at the boundary between the Network layer and the Transport layer.

1.4.1.7 Network-Service-Access-Point

The point at which network-services are provided by a network-entity to a transport-entity.

1.4.1.8 Network-Service-Access-Point-Address

An identifier which tells where a network-service-access-point may be found.

1.4.1.9 Routing

A function within a layer which translates the title of an entity or the service-access-point-address to which the entity is attached into a path by which the entity can be reached.

1.4.1.10 Sublayer

A subdivision of a layer.

1.4.1.11 Subnetwork

A set of one or more intermediate systems which provide relaying and through which end systems may establish network-connections.

1.4.2 Additional Definitions

For the purpose of this Standard, the following definitions apply:

1.4.2.1 Automaton

A machine designed to follow automatically a predetermined sequence of operations or to respond to encoded instructions.

1.4.2.2 End System

An Open System which contains all seven layer of the Open Systems Interconnection archtecture.

1.4.2.3 Internetwork Protocol

A subnetwork Independent Convergence Protocol combined with relay and routing functions.

1.4.2.4 Internetwork Protocol Data Unit

Data unit exchanged between entities implementing this Standard.

1.4.2.5 Internetwork Protocol Data Unit Segment

Data unit resulting from segmentation of an original internetwork protocol data unit.

1.4.2.6 Subnetwork Service

The set of functions provided by a subnetwork.

1.4.2.7 Subnetwork Service Access Point

The point at which subnetwork services are provided by a subnetwork entity.

1.5 Acronyms

1.5.1 Data Units

IPDU	Internetwork Protocol Data Unit
NSDU	Network Service Data Unit
SDSDU	Subnetwork Dependent Service Data Unit

1.5.2 Internetwork Protocol Data Unit

DT IPDU	Data Internetwork Protocol Data Unit

1.5.3 IPDU Fields

NLPI	Network Layer Protocol Identification
LI	Length Indicator
V/PE	Version/Protocol Identifier Extension
LT	Lifetime
SP	Segmentation Permitted Flag
MS	More Segments Flag
TP	Type
SL	Segment Length
DAL	Destination Address Length
DA	Destination Address
SAL	Source Address Length
SA	Source Address
DUID	Data Unit Identifier
SO	Segment Offset
TL	Total Length

1.5.4 Parameters

DA	Destination Address
SA	Source Address
QOS	Quality of Service

1.5.5 Miscellaneous

SNICP	Subnetwork Independent Convergence Protocol
SNDCP	Subnetwork Dependent Convergence Protocol
SNAP	Subnetwork Access Protocol
SN	Subnetwork
IP	Internetwork Protocol
IPCI	Internetwork Protocol Control Information
NS	Network Service
N	Network

2 Protocol

2. PROTOCOL

2.1 Overview of the Protocol

2.1.1 Rationale

The Connectionless Internetwork protocol provides a connectionless network service, as described in 2.1.5. The basis for chosing either a connectionless or connection-oriented network service is outside the scope of this Standard; however, the following aspects of the connectionless network service may be taken into account:

- connectionless operation is highly tolerant of faults occurring in the supporting subnetworks;
- connectionless operation is highly tolerant of faults occurring in internetwork gateways and other network layer components not comprising part of the supporting subnetworks;
- connectionless operation may be appropriate when the configuration of the total network is subject to frequent or unpredictable change, as when using dynamic routing techniques.

2.1.2 Internal Organization of the Network Layer

The architecture of the Network layer is described in a separate document, ISO/TC97/SC6 N2613 (see 1.3), where a target OSI Network layer structure is defined, and a structure to classify protocols as an aid to the progression toward that target structure is presented. The Internetwork protocol herein described is a Subnetwork Independent Convergence protocol combined with relay and routing functions designed to allow the incorporation of existing network Standards within the OSI framework.

A Subnetwork Independent Convergence protocol is one which can be defined on a subnetwork independent basis and which is necessary to support the uniform appearance of the connectionless OSI Network Service over a set of interconnected heterogeneous subnetworks. The Connectionless Internetwork protocol is defined in such a subnetwork independent way so as to minimize variability where subnetwork dependent and/or subnetwork access protocols do not provide the OSI Network service.

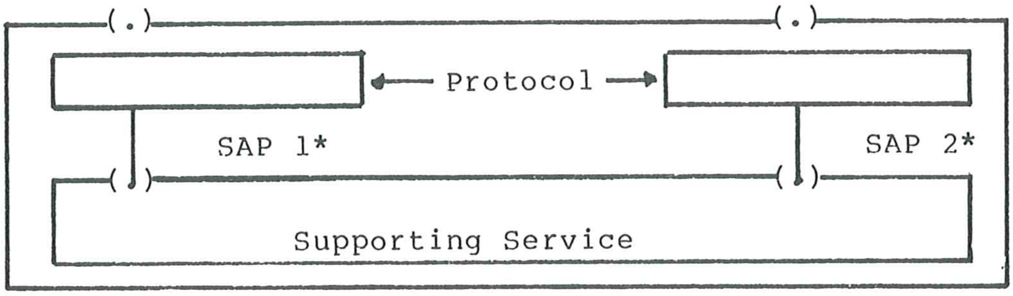
The subnetwork service required for the lower sublayers by the Internetwork protocol is identified in 2.1.5. This service may be provided either directly by a subnetwork, or by a Subnetwork Dependent Convergence protocol.

2.1.3 Model of Interworking

Two distinct types of models have been identified in the internal organization of the Network layer as necessary to represent the specifications needed in the Network layer. This document employs the protocol model therein described.

2.1.3.1 Protocol Model

A protocol is any set of rules by which the interaction of two or more entities is governed. Such rules include the association of specific meaning with lower layer service primitives and parameters, and the establishment of a priori agreements between cooperating peer entities. The operation of a protocol is modelled by exchanges between pairs of entities, using a supporting service, as illustrated below:



*) The same service is seen at SAP 1 as at SAP 2

Fig. 2 Operation of a Protocol

2.1.3.2 Application of the Protocol Model

As stated in ISO/TC97/SC6 N2613 (see 1.3), the general modelling for protocols described in the internal organization of the Network layer may be applied to the real world in whatever way is indicated by the designer's choice of system boundaries. An illustration of how the model applies to the Internetwork protocol follows.

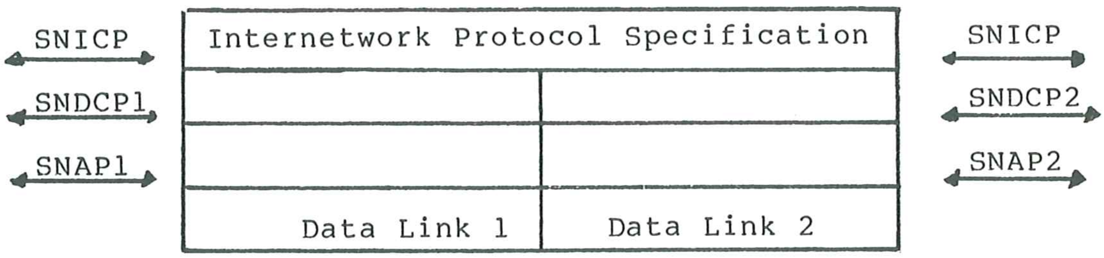


Fig. 3 Application of the Protocol Model

In this example, the specification of the Subnetwork

Independent Convergence protocol is combined with the specification of relay and routing functions. The format of the Subnetwork Independent Convergence Protocol Data Unit transmitted by an intermediate system is the same as that of the Subnetwork Independent Convergence Protocol Data Unit it has received.

2.1.4 Addressing

The Source address and the Destination address parameters referred to in 2.4.4 are OSI Network Addresses. The precise nature and length of these addresses are not defined in this Standard.

2.1.5 Service Provided by the Network Layer

The service provided by the protocol herein described is a Connectionless Network service.

The Connectionless Network service is described in document ISO/TC97/SC16 N2713 (see 1.3). The network service primitives provided are summarized in Table 1.

Primitives	Parameters
N_UNITDATA Request Indication	NS_Destination_Address NS_Source_Address NS_Quality_of_Service NS_Userdata

Table 1 Network Service Primitives

It is not a requirement that a connectionless mode service should support an unbounded NSDU size. This Internetwork protocol supports a maximum length of 65535 octets for an NSDU. When the subset for LAN is used, the maximum NSDU length is 1024 octets less the amount used for the IPDU header.

2.1.5.1 Network Quality of Service

Network quality of service refers to certain characteristics of connectionless mode transmission as observed between the service access points. Quality of Service describes aspects of a connectionless mode transmission which are attributable solely to the Network Service Provider. The quality of service parameters identified for the network service are:

- transit delay
- protection against unauthorized access
- cost determinants
- maximum NSDU lifetime.

2.1.6 Service Assumed from Subnetwork Service Provider

The subnetwork service required to support the Connectionless Internetwork protocol is defined as comprising the primitives defined in Table 2.

Primitives	Parameters
SN_UNITDATA Request Indication	SN_Destination_Address SN_Source_Address SN_Quality_of_Service SN_Userdata

Table 2 Subnetwork Service Primitives

2.1.6.1 Subnetwork Addresses

The Source and Destination addresses specify the subnetwork service access points involved in the transmission.

The precise nature and exact length of subnetwork addresses are not defined in this Standard.

2.1.6.2 Subnetwork Quality of Service

Subnetwork quality of service describes aspects of a subnetwork connectionless mode service which are attributable solely to the subnetwork service provider.

Associated with each subnetwork connectionless mode transmission, certain measures of quality of service are agreed upon when the primitive action is initiated. The requested measures (or parameter values and options) are based on an a priori knowledge by the Network service provider of the service(s) made available to it by the subnetwork. Knowledge of the nature and type of service available is typically obtained through some implementation-specific primitive action prior to any invocation of the subnetwork connectionless mode service.

The quality of service parameters identified for the subnetwork connectionless mode service are:

- transit delay
- protection against unauthorized access
- cost determinants

2.1.6.3 Subnetwork User Data

The Subnetwork User Data (SN_Userdata) is an ordered multiple of octets, and is transferred transparently between the specified subnetwork service access points.

The Subnetwork service is required to support a subnetwork service data unit size of at least the size of the IPDU header plus one octet of NS_Userdata. When the subset for LAN is in use, the Subnetwork service is required to support an SNSDU size of at least 1024 octets.

2.1.6.4 Subnetwork Dependent Convergence Functions

Subnetwork Dependent Convergence functions may be performed to provide connectionless Subnetwork service in the case where subnetworks provide a connection-oriented service. If a subnetwork provides the connection-oriented service, some subnetwork dependent function is assumed to provide a mapping into the required subnetwork service described in the preceding text.

A Subnetwork Dependent Convergence protocol may also be employed in those cases where functions assumed from the Subnetwork service provider are not provided.

2.2 Protocol Functions

This section serves for explanation of the protocol functions only. It does, however, not impose any rule or restrictions on protocol implementations.
The functions described in the following subsections are supplied by this Internetwork protocol.

2.2.1 IPDU Composition Function

This function is responsible for the construction of an IPDU according to the rules of protocol given in 2.4.

Protocol Control Information required for delivering the data unit to its destination is determined from current state information and from the parameters provided with the N_UNITDATA Request; e.g. Source and Destination addresses, QOS, etc. User data passed from the Network service user in the N_UNITDATA Request form the Data Field of the IPDU.

During the composition of the IPDU, a Data Unit Identifier is assigned to uniquely identify all segments (there may be only one) of NS_Userdata from a particular service data unit. This identifier may be also used for ancillary functions such as error reporting. The "Reassemble IPDU" function judges segments to belong to the same original SDU, hence IPDU, if they have the same Source and Destination addresses and Data Unit Identifier. The originator of the IPDU must choose the Data Unit Identifier so that it remains unique for this Source/Destination address pair and protocol for the maximum lifetime of the IPDU, or any segment thereof, in the network.

2.2.2 IPDU Decomposition Function

This function is responsible for stripping off the Internetwork Protocol Control Information from the IPDU. During this process, information pertinent to the generation of the N_UNITDATA Indication is retained. The data field of the IPDU received is reserved until all segments of the original service data unit have been received; this is the NS_Userdata Parameter of the N_UNITDATA Indication.

2.2.3 Header Format Analysis Function

This function determines which Internetwork protocol Data Unit Header Format is employed. If the IPDU has a non-zero length IPDU header, then this function determines whether an IPDU received has reached its destination using the Destination address provided in the IPDU. If the Destination address provided in the IPDU is the same as the one which addresses a transport entity served by this network entity, then the IPDU has reached its destination; if not, it must be forwarded. If the IPDU has a zero length IPDU header (see 2.4.9), then the destination has been reached.

2.2.4 IPDU Lifetime Control Function

Closely associated with the header format analysis function, this function determines whether an IPDU received may be forwarded or whether its assigned lifetime has expired, in which case it must be discarded.

2.2.5 Routing and Forwarding Function

This function analyses the Destination NSAP address, quality of service and/or other parameters and is able to determine:

- whether this Destination address corresponds to the local NSAP address so that the user data is considered to have arrived at its destination;

- whether the destination domain ID corresponds to the local subnetwork but the destination domain-dependent ID does not correspond to a local NSAP, and then determines the SNAP address that has to be offered to the subnetwork to identify the end system to which the IPDU has to be forwarded;

- the subnetwork that has to be chosen to access the subsequent gateway to which an IPDU has to be forwarded;

- the SNAP address that has to be offered to that subnetwork so as to identify the subsequent gateway.

It has to update the header information of IPDUs to be forwarded.

NOTE 1

A routing management function is responsible for controlling the operation of the Routing and Forwarding function.

2.2.6 Segment IPDU Function

Segmentation is performed when the size of the Internetwork Protocol Data Unit is greater than the maximum size of the user data parameter field of the subnetwork service primitive.

Segmentation consists of composing two or more new IPDUs from the IPDU received. The IPCI required to identify, route and forward an IPDU is duplicated. The user data encapsulated within the IPDU received is divided in such a way that the new IPDUs satisfy the size requirements of the user data parameter field of the subnetwork service primitive.

IPDU segments are identified as being from the same original IPDU by means of

- the Source address;
- the Destination address;
- the data unit identifier.

A segment field offset field identifies where (i.e. which octet) in the data field of the original IPDU the segment begins. A Segment Length field specifies the length in octets of the IPDU segment. A More Segments flag is set to ONE if this segment is not the last segment, and is set to ZERO if this segment is the last one. A Total Length field specifies the entire length of the IPDU (before segmentation) including both header and data, if present. IPDU segments may be further segmented without constraining the routing of the individual segments.

A Segmentation Permitted flag is set to ONE to indicate that segmentation is permitted. If the original IPDU is not to be segmented at any (further) point during its lifetime in the network, the flag is set to ZERO. When the Segmentation Permitted flag is set to ZERO, the Segment Length specifies the entire length of the IPDU segment, including both header and data, if present.

2.2.7 Reassemble IPDU Function

Reassembly of the IPDU must be performed prior to the IPDU Decomposition function (see 2.2.2).

Reassembly consists of reconstructing the original IPDU transmitted by the destination internetwork entity from the segment(s) generated during the lifetime of the original data unit.

NOTE 2

Internetwork segmentation based solely on knowledge of maximum SDU sizes of adjacent subnetworks requires that the IPDU be reassembled at the destination. Other segmentation schemes which:
- interact with the routing algorithm to favour path on which fewer segments are generated;
- generate more segments than absolutely required in order to avoid additional segmentation at some subsequent point; or
- allow partial/full reassembly at some point along the route where it is known that the subnetwork with the smallest IPDU size has transited;

are not precluded. The information necessary to enable the use of one of these alternative strategies may be made available through the operation of an Internetwork Management Function. The exact nature of this management function is for further study.

2.2.8 Discard IPDU Function

This function performs all of the actions necessary to free the resources reserved by the network-entity in any of the following situations (note that the list is not exhaustive):

- an IPDU is received whose header cannot be analyzed;
- an IPDU is received whose lifetime has expired;
- segment(s) of an IPDU are being held at a reassembly point, and the reassembly lifetime assigned to that IPDU expires;
- an IPDU is received which cannot be segmented and cannot be forwarded because its length exceeds the maximum subnetwork service data unit size;
- an IPDU is discarded for the purpose of relieving congestion.

NOTE 3

With respect to the last item, a requirement has been identified for a congestion control function. The mechanism for providing this function are for further study.

2.2.9 IPDU Error Detection Function

This function protects network entities against possible failure or malfunction due to the processing of erroneous information in the IPDU header.

A mechanism to provide this function is not specified by this version of this Internetwork protocol. However, the possible future need for this function is recognized. To permit compatible use of a version of this protocol including such a mechanism, a two octet field is included in the IPDU header, with the designation: "Reserved for IPDU Header Error Detection Function".

2.2.10 Optional Functions

Optional functions are functions that may or may not be supported by an individual comforming implementation of the Internetwork protocol. Optional functions will comprise the Options part of the IPDU header.
No options are defined in this version of the Standard.

2.3 Description of the Internetwork Protocol

This section explains only the protocol functions and does not impose any rules or restrictions on protocol implementations.

The interrelationship of some of the Internetwork protocol functions described in 2.2 can be represented as in Fig. 4.

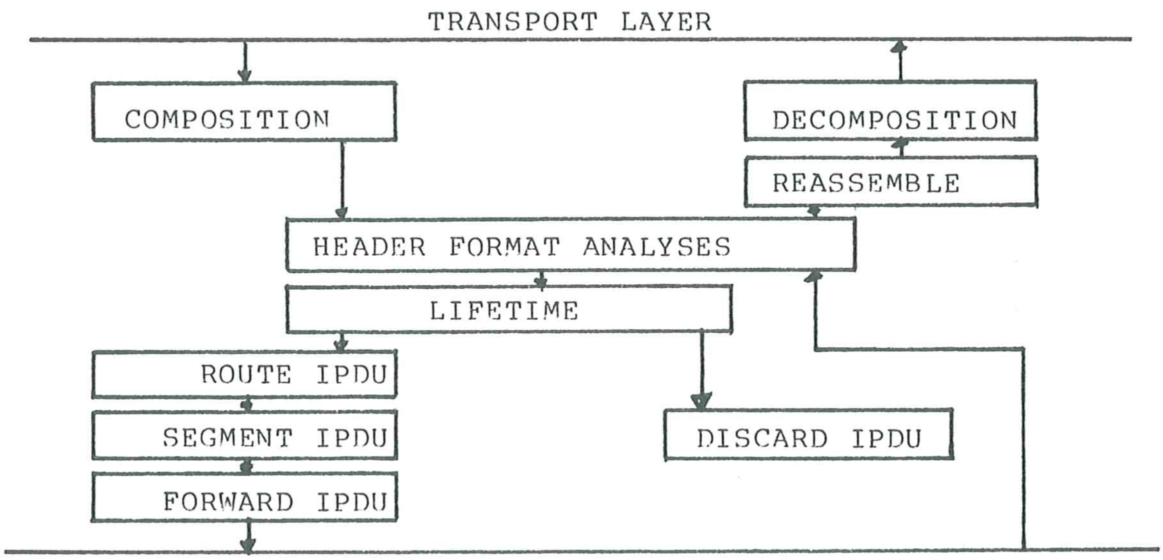


Fig.4 Subnetwork Dependent Service

2.4 Structure and Encoding of IPDUs

2.4.1 Structure

All the Internetwork Protocol Data Units (IPDUs) shall contain an integral number of octets. The octets in an IPDU are numbered starting from one and increasing in the order in which they are put into an SNSDU. The bits in an octet are numbered from 1 to 8, where bit 1 has the least significant value.

When consecutive octets are used to represent a binary number, the octet with the lowest number has the least significant value.

When the encoding of an IPDU is represented using a diagram in this section, the octets are shown with bit 8 to the left and bit 1 to the right of the diagram.

IPDUs shall contain, in the following order:

- the Header, comprising:

  - the Network Layer Protocol Identification part;
  - the fixed part;
  - the address part;
  - the segmentation part, if present;
  - the options part, if present;

and

- the Data Field, if present.

This structure is illustrated in Fig. 5.

Part	Described in
Network Layer Protocol Ident. Part	2.4.2
Fixed Part	2.4.3
Address Part	2.4.4
Segmentation Part	2.4.5
Options Part	2.4.6
Data	2.4.7

Fig. 5 IPDU Structure

2.4.2 Network Layer Protocol Identification Part

This part contains information which allows the Connectionless Internetwork protocol to be distinguished uniquely from other protocols used in the Network layer of OSI. It is illustrated in Fig. 6.

bit 8 bit 1	Octet
Network Layer Protocol Ident.	1

Fig. 6 IPDU Header - Network Layer Protocol Ident. Part

2.4.2.1 Network Layer Protocol Identification

The value of this field is binary 1000 0001. This is the value assigned by ISO to identify the Connectionless Internetwork protocol.

2.4.3 Fixed Part

2.4.3.1 General

The fixed part contains frequently occurring parameters including the code of the IPDU. The length and the structure of the fixed part are defined by the IPDU code.

The fixed part is illustrated in Fig. 7.

bit 8		bit 1	Octet
Length Indicator			1
Version/Protocol ID			2
Lifetime			3
SP	MS	Type	4
Segment Length			5 6
Reserved for use by IPDU Header Error Detection Function			7 8

Fig. 7 IPDU Header - Fixed Part

2.4.3.2 Length Indicator

This field is contained in the first octet of all IPDUs. The length is indicated by a binary number, with a maximum value of 254 (1111 1110). The length indicated is the header length in octets, including parameters, but excluding the length indicator field and user data, if any. The value 255 (1111 1111) is reserved for possible extensions.

2.4.3.3 Version/Protocol ID Extension

The value of this field is binary 0000 0001. This field identifies version 1 of this Internetwork protocol.

2.4.3.4 IPDU Lifetime

The Lifetime field is encoded as a binary number that limits the time that the IPDU may remain in the network. The original value of this field is established by the source network entity. The

Lifetime field is decremented by each of the network-entities which subsequently process the IPDU according to the methods identified below. If the Lifetime field reaches a value of ZERO before the IPDU is delivered to the destination network-entity, the IPDU will be discarded.

Network-entities processing an IPDU shall always decrement the Lifetime field by at least one. They should decrement the Lifetime field by more than one if the sum of the transit delay in the subnetwork from which the IPDU was received and the delay within the system processing the IPDU exceeds or is estimated to exceed 500 ms.

Under these circumstances the Lifetime field should be decremented by one for each additional 500 ms of delay. While the determination of delay need not to be precise,it is recognized that overestimates are preferable to underestimates, since underestimates could defeat the purpose of maintaining a Lifetime field.

2.4.3.5 Segmentation Permitted and More Segments Flags

The Segmentation Permitted (SP) flag determines whether segmentation is permitted. A value of ONE indicates that segmentation is permitted, a value of ZERO indicates that it is not.

When the Segmentation Permitted flag is set to ONE, the More Segments (MS) flag indicates whether the data unit identified by the Data Unit Identifier field in the segmentation part of the IPDU header has been segmented. When the More Segments flag is set to ONE, then the Segment Offset field in the segmentation part of the IPDU header indicates where in the data field of the original IPDU this segment begins.

When the More Segments flag is set to ZERO, and the Segmentation Permitted flag is set to ONE, then this is the last segment of the original IPDU, and the Segment Offset field again indicates where in the data field of the original IPDU this final segment begins.

When the Segmentation Permitted flag is set to ZERO, the More Segments flag shall also be set to ZERO.

When the Segmentation Permitted flag is set to ZERO, the segmentation part of the IPDU header is not present.

2.4.3.6 Type Code

The type Code field identifies the type of the protocol data unit. The allowed value is given in Table 3.

DT	DATA	111100

Table 3 Valid IPDU Type

2.4.3.7 IPDU Segment Length

The Segment Length field specifies the entire length of the IPDU segment including both header and data, if present.

This field contains a binary number, with octets ordered as described in 2.3

For unsegmented IPDUs it should be noted that the value of this field is identical to the value of the Total Length field located in the Segmentation Part of the header, if present.

2.4.3.8 Octets 7 and 8

These octets are reserved for future definition of an IPDU Header Error Detection Mechanism.

2.4.4 Address Part

Address parameters are distinguished by their location, immediately following the fixed part of the IPDU header. The address format is illustrated in Fig. 8.

bit 8 bit 1	Octet
Destination Address Length Indicator	9
Destination Address	10 m-1
Source Address Length Indicator	m
Source Address	m+1 n-1

Fig. 8 IPDU Header - Address Part

The Destination and Source Address are network service access point addresses as defined in ECMA-TR/20 (see 1.3). The Destination Address Length Indicator field is a binary number which specifies the length of the Destination Address in number of octets. The Destination Address field follows the Destination Address Length Indicator field.

The Source Address Length Indicator field is a binary number which specifies the length of the Source Address in number of octets. The Source Address Length Indicator field follows the Destination Address field. The Source Address field follows the Source Address Length Indicator field.

2.4.5 Segmentation Part

If the Segmentation Permitted flag in the Fixed Part of the IPDU header (octet 4, bit 8) is set to ONE, the segmentation part of the header illustrated below shall be present.

bit 8 bit 1	Octet
Data Unit Identifier	n n+1
Segment Offset	n+2 n+3
Total Length	n+4 n+5

Fig. 9 Segmentation Part

2.4.5.1 Data Unit Identifier

The Data Unit Identifier identifies to which data unit a segment belongs so that a segmented data unit may be correctly reassembled by the destination network-entity.

Values are serially assigned and may wrap around. The Data Unit Identifier is a binary number whose size is two octets.

2.4.5.2 Segment Offset

For each segment the Segment Offset field specifies the relative position of the segment in the original, complete data unit with respect to the start of the data field. The offset is measured in octets. The offset of the first segment is zero. The Segment Offset is a binary number.

2.4.5.3 IPDU Total Length

The Total Length field is a binary number which specifies the entire length of the IPDU header and data, if present.

2.4.6 Options Part

The options part is used to define optional parameters. If the option part is present, it shall contain one or more parameters. The number of parameters that may be contained in the option part is indicated by the length of the option part which is: Length Indicator - (length of fixed part + length of address part + length of segmentation part).

Each parameter contained within the options part of the IPDU header is encoded as described in Figure 1Ø.

bit 8 bit 1	Octet
Parameter Code	n
Parameter Length (m)	n+3
Parameter Value	n+2 n+2+m

Fig. 1Ø Encoding of Parameters

The Parameter Code field is coded in binary and, without extensions, provides a maximum number of 255 different parameters. However, as noted below, bits 8 and 7 cannot take every possible values, so the practical maximum number of different parameters is less than 255. A Parameter Code of all ONE (1111 1111) is reserved for possible extensions of the Parameter Code.

The Parameter Length field indicates the length, in octets, of the parameter value field. The length is indicated by a binary number "m", with a maximum theoretical value of 255. The practical maximum value of m is lower. For example, in the case of a single parameter contained within the variable part, two octets are required for the Parameter Code and the Parameter Length indication itself. Thus the value of "m" is limited to: 253 - (length of fixed part + length of address part + length of segmentation part). For larger fixed parts of the header and for each succeeding parameter the maximum value of "m" decreases.

The parameter value field contains the value of the parameter identified in the Parameter Code field.

No Parameter Codes shall use bits 8 and 7 with the value ZERO.

Implementations shall accept the parameters defined in the options part in any order, providing that they are processed according to their grouping. If any parameter is duplicated, then the later value will be used.

No optional parameters are defined in this version of the Standard.

2.4.7 Data Part

The Data part of the IPDU is structured as an ordered multiple of octets, which is identical to the same ordered multiple of octets specified in the NS-Userdata parameter of the N_UNITDATA Request and Indication primitives.

The data field is illustrated in Fig. 11.

bit 8 bit 1	Octet
Data	p+1 z

Fig. 11 IPDU Data Field

2.4.8 Data (DT) IPDU

2.4.8.1 Structure

The DT IPDU shall have the structure defined in Fig. 12.

bit 8 bit 1	Octet
Network Layer Protocol Ident.	1
Length Indicator	2
Version/Protocol ID	3
Lifetime	4
SP MS Type	5
Segment Length	6 7
Reserved for use by IPDU Header Error Detection Function	8 9
Destination Address Length Indicator	10
Destination Address	11 m-1
Source Address Length Indicator	m
Source Address	m+1 n-1
Data Unit Identifier	n n+1
Segment Offset	n+2 n+3
Total Length	n+4 n+5
Options	n+6 p
Data	p+1 z

Fig. 12 IPDU - Variable Length Header Format

2.4.8.2 Network Layer Protocol Identification Part

See 2.4.2

2.4.8.3 Fixed Part

- Length Indicator	see 2.4.3.2
- Version/Protocol ID	see 2.4.3.3
- Lifetime	see 2.4.3.4
- SP, MS	see 2.4.3.5
- Type	see 2.4.3.6
- Segment Length	see 2.4.3.7
- IPDU Header Error Indicator Function	see 2.4.3.8

2.4.8.4 Addresses

See 2.4.4

2.4.8.5 Options

See 2.4.6

2.4.8.6 Data

See 2.4.7

2.4.9 Zero Length Header IPDU

2.4.9.1 Structure

The structure of the zero length header IPDU shall be as defined in Fig. 13:

bit 8 ... bit 1	Octet
Length Indicator	1
Data	2 n

Fig. 13 IPDU - Zero Length Header

2.4.9.2 Network Layer Identification Field

This field contains the value ZERO (0000 0000). This is the value assigned by ISO to distinguish the Inactive Network layer protocol (i.e. the absence of explicit Protocol Control Information for the Network layer) from other Network layer protocols.

2.4.9.3 Data Field

See 2.4.7. The NS_Userdata parameter is constrained to be less than or equal to the value of the SN_Userdata parameter minus ONE.

3 Interconnection of LAN

3. INTERCONNECTION OF LOCAL AREA NETWORKS

3.1 Introduction to Local Area Network Requirements

Within the scope of this Standard there is an identified need for a simplified Connectionless Internetwork protocol, suitable for the interconnection of Local Area Networks (LAN) either directly or by simple Wide Area Network Connection. Examples of such interconnections are illustrated in Figures 14 to 16.

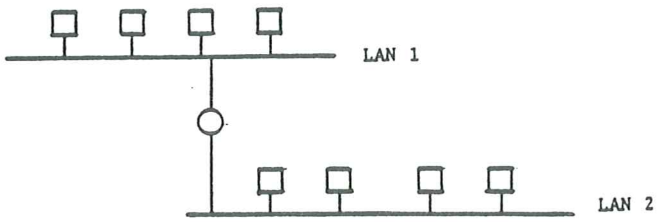


Fig. 14 Direct Interconnection of two LAN

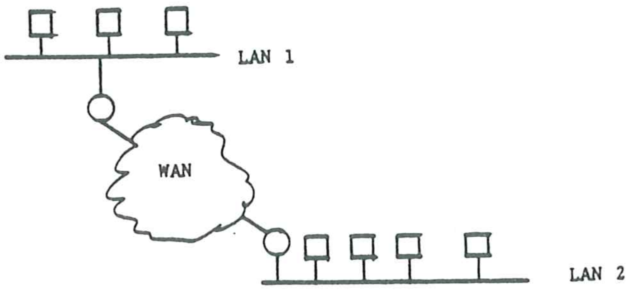


Fig. 15 Interconnection of two LAN by Single WAN

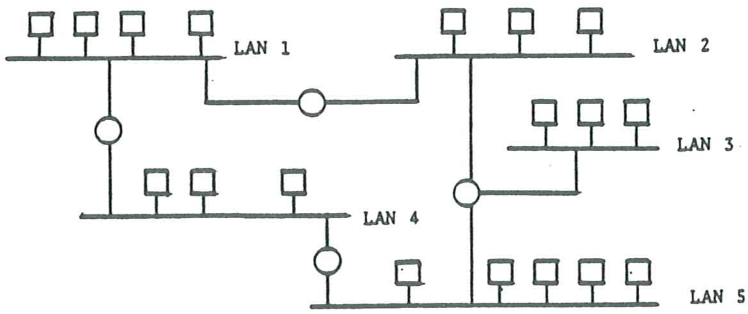


Fig. 16 Interconnection of Multiple LAN

3.2 Technical Description

For the applications described in 3.1, some assumptions may be made concerning the available Subnetwork service, which permit use of a subset of the Connectionless

Internetwork protocol and still provide a complete connectionless network service.
In particular:

- a maximum SNSDU size is available which is large enough to avoid the need for segmentation;
- a uniform subnetwork quality of service is available;
- the topology of the internetwork is simple enough not to require complex routing decisions.

The subset of the Connectionless Internetwork protocol to be used in this application does not use the following functions described in 2.2 of this Standard:

- Segmentation and Reassembly (see 2.2.6 and 2.2.7);
- IPDU Header Error Detection (not included in this Standard);
- IPDU Options.

3.3 IPDU Format

The format of the IPDU header in this case is as shown in Fig. 17.

bit 8 … bit 1	Octet
Network Layer Protocol Ident.	1
Length Indicator	2
Version/Protocol ID	3
Lifetime	4
SP | MS | Type	5
IPDU Length	6 7
Reserved for use by IPDU Header Error Detection Function	8 9
Destination Address Length Indicator	10
Destination Address	11 22
Source Address Length Indicator	23
Source Address	24 35
Data	36 z

Fig.17 IPDU - Fixed Length Header Format

3.4 Format

3.4.1 Fixed Fields

- Network Layer Prot. Id.	set to decimal value 129
- Length Indicator	set to decimal value 34
- Version/Protocol ID	see 2.4.3.3
- Lifetime	see 2.4.3.4. May be set initially to 1111 1110 for this subset
- SP, MS	SP= 0, MS = 0
- Type	111100
- Segment Length	see 2.4.3.7
- IPDU Header Error Detection function	see 2.4.3.8. Shall be set to ZERO for this subset

3.4.2 Addresses

- Destination Address Length	set to decimal value 12
- Source Address Length	set to decimal value 12

See also 2.4.4

3.4.3 Segmentation and Options

The Segmentation and Options parts of the IPDU header are not present in this DT IPDU variant.

3.4.4 Data

See 2.4.7. The NS_Userdata parameter is constrained to be less than or equal to the value of the SN_Userdata parameter, minus 33.

3.5 IPDU Lifetime Control

For the application described in 3.1, the likelihood of excessive IPDU lifetime is greatly reduced. Therefore a simpler lifetime control mechanism may be used. The IPDU lifetime field of the IPDU must be decremented by ONE whenever the IPDU is forwarded, but the timer rule does not need to be applied. The initial value of the field may be set to 254 to avoid the need for the originator of the IPDU to have any knowledge of network topology, subnetwork transit delay and other relevant factors.

4 Formal Description of the Protocol

4. FORMAL DESCRIPTION OF THE PROTOCOL

4.1 Introduction

This section contains a formal description of the Connectionless Internetwork protocol, modelled as a finite state automaton, governed by a state variable with three values. The behaviour of the automaton is defined with respect to individual, independent IPDUs. This is a consequence of the connectionless nature of the protocol. The operation of the automaton is defined by the use of the formal description techniques and notation specified in ISO/TC97/SC16 N1347 (see 1.3). This technique is based on an extended finite state transition model. It uses the Pascal programming language, with extensions to permit the description of the extended finite state automaton. This specification formally specifies an abstract machine which provides the abstract connectionless network service definition by use of the Connectionless Internetwork protocol. It should be emphasized that this formal specification does not in any way constrain the internal operation or design of any actual implementation. For example, it is not anticipated that the program segments contained in the state transitions will actually appear as part of an actual implementation. A formal protocol specification is useful in that it does as far as possible eliminate any degree of ambiguity or vagueness in the specification of a protocol Standard.

4.2 Values of the State Variables

The Connectionless Internetwork protocol state variable has three values:

INITIAL	The automaton is created in the INITIAL state. No transition may bring the automaton into the INITIAL state.
REASSEMBLING	The automaton is in the REASSEMBLING state for the period in which it is reassembling IPDU segments into a complete IPDU.
CLOSED	The final state of the automaton is the CLOSED state. When the automaton enters the CLOSED state it ceases to exist.

4.3 Type and Constants Definition

```
const
        empty        =   Ø;
        ZERO         =   Ø;
        null         =   Ø;
        CL_IP_nlpi = 129;

type
        NSAP_addr_type  = ...;
        NPAI_addr_type  = ...;
        SN_addr_type    = ...;

        quality_of_service_type = ...;
        SN_QOS_type     = ...;

        data_type       = ...;
        buffer_type     = ...;

        integer_type    = ...;
        timer_name_type = (lifetime_timer);
        timer_data_type = ...;
        version_id_type = (CL_IP_version);
        boolean_type    = (FALSE, TRUE);
        ipdu_tp         = (DT);
        options_type    = ...;
        subnet_id_type  = ...;
        result_type     = (FAILURE, SUCCESS);
        error_type      = (null,
                           DESTINATION_UNREACHABLE,
                           DESTINATION_UNKNOWN,
                           LIFETIME_EXPIRED,
                           CONGESTION,
                           IPDU_HEADER_ERROR,
                           SEG_NEEDED_AND_NOT_PERMITTED,
                           PROTOCOL_ERROR);

nsdu_type      = record
                     da   : NSAP_addr_type;
                     sa   : NSAP_addr_type;
                     qos  : quality_of_service_type;
                     data : data_type;
                 end;

ipdu_type      = record
                     nlpi     : integer_type
                     hli      : integer_type;
                     vp_id    : version_id_type;
                     lifetime : integer_type;
                     sp       : boolean_type;
                     ms       : boolean_type;
                     ipdu_tp  : ipdu_tp_type;
                     seg-len  : integer_type;
                     reserved : integer_type;
                     da-len   : integer_type;
```

```
                    da         : NPAI_addr_type;
                    sa_len     : integer_type;
                    sa         : NPAI_addr_type;
                    du-id      : integer_type;
                    so         : integer_type;
                    tot_len    : integer_type;
                    options    : integer_type;
                    data       : data_type;
                end;

sn_route_type = record
                    subnet-id : subnet-id_type;
                    sn_da     : SN_addr_type;
                    sn_sa     : SN_addr_type;
                    result    : result_type;
                    error     : error_type;
                end;
```

4.4 Interface Definition

```
channel Network_access_point (User, Provider);

   by User:
       UNITDATA_request
         (NS_Destination_address  : NSAP_addr_type;
          NS_Source-address       : NSAP_addr_type;
          NS_Quality_of_service   : quality_of_service_type;
          NS_Userdata             : data_type);

   by Provider:
       UNITDATA_indication
         (NS_Destination-address  : NSAP_addr_type;
          NS_Source-address       : NSAP_addr_type;
          NS_Quality_of_service   : quality_of_service_type;
          NS_Userdata             : data_type);

channel Subnetwork_access_point (User, Provider);

   by User:
       UNITDATA_request
         (SN_Destination-address  : SN_addr_type;
          SN_Source-address       : SN_addr_type;
          SN_Quality_of_service   : SN_QOS_type;
          SN_Userdata             : ipdu_type);

   by Provider:
       UNITDATA_indication
         (SN_Indication-address   : SN_addr_type;
          SN_Source-address       : SN_addr_type;
          SN_Quality_of_service   : SN_QOS_type;
          SN_Userdata             : ipdu_type);
```

```
channel system_access_point (User, Provider);

   by User:
       TIMER_request
         (Time      : integer_type;
          Name      : timer_name_type;
          Datum     : timer_data_type);

       TIMER_cancel
         (Name      : timer_name_type);

   by Provider:
       TIMER_indication
         (Name      : timer_name_type;
          Datum     : timer_data_type);
```

4.5 Formal Machine Description

```
module IP_Machine
     (N  : Network_access_point (Provider) common queue;
      SN : Subnetwork_access_point (User) common queue;
      S  : System_access_point (User) individual queue;

var
      nsdu    : nsdu_type;
      ipdu    : ipdu_type;
      rcv_buf : buffer_type;

state  : (INITIAL, REASSEMBLING, CLOSED);

initialize:
      begin
          state to INITIAL;
          rcv_buf := empty;
      end;

procedure send_ipdu (ipdu : ipdu_type);

var
  snr          : sn_route_type;
  max_data     : integer_type;
  data_buf     : buffer_type;
  more_seg     : boolean_type;
  sn_qos       : SN_QOS_type;
  size_to_send : integer_type;

begin
  snr := route (ipdu);

  if (snr.result = SUCCESS) then
    begin
      max_data := sn_data_maxsize (snr.subnet_id)-ipdu.hli;

      if (max_data =< ZERO then
              release_ipdu (PROTOCOL_ERROR, ipdu);
```

```
      else if (max_data <size (ipdu.data)) and
                          (not ipdu.sp)    then
               release_ipdu (SEG_NEEDED_AND_NOT_PERMITTED,
                          ipdu);

else
  begin
    more_seg := ipdu.ms;
    data_buf := make_buffer(ipdu.data);
    repeat
      begin
        size_to_send := select_segment_size (min (max_data,
                          size_buf (data_buf)), ipdu);

        ipdu.data    := extract (data_buf, size_to_send);

        ipdu.seg_len := ipdu.hli + size (ipdu.data);

        if (size_buf (data_buf) = ZERO) then
          ipdu.ms  := more_seg;

        else
          ipdu.ms  := true;

        sn_qos   := get_sn_qos (snr.subnet_id);

        out SN[snr.subnet_id].UNITDATA_request
                (snr.snda, snr.snsa, sn_qos, ipdu);

        ipdu.so   :=  ipdu.so + size_to_send;
       end;
     until (size_buff (data_buf) = ZERO);
    end;
   end;

 else if (ipdu.ipdu_tp = DT) then
    begin
      release_ipdu (sn.error, ipdu);
    end;

end;


procedure allocate_reassembly_resources
     (ipdu_tot_len : integer_type);
primitive;

function data_unit_complete
      (buf : buffer_type) : boolean_type;
primitive;

procedure decrement_lifetime
      (lifetime : integer_type);
primitive;
```

```
function extract
       (buf    : buffer_type;
        amount : integer_type) : data_type;
primitive;

function get_data_unit_identifier = integer_type;
primitive;

function release_ipdu
       (error : error_type;
        ipdu  : ipdu_type) : data_type;
primitive;

function get_header_len
       (da_len  : integer_type;
        sa_len  : integer_type;
        sp      : boolean_type;
        options : options_type) : integer_type;
primitive;

function get_lifetime
       (da      : NSAP_addr_type;
        qos     : quality_of_service_type : lifetime_type;
primitive;

function get_local_NPAI_addr : NPAI_address_type;
primitive;

function get_local_NPAI_addr_len : integer_type;
primitive;

function get_NPAI
       (addr : NSAP_addr_type) : NPAI_addr_type;
primitive;

function get_NPAI_len
       (addr : NSAP_addr_type) : integer_type;
primitive;

function  get_NSAP_addr
       (addr    : NPAI_addr_type;
        ler     : integer_type) : NSAP_addr_type;
primitive;

function get_seg_permitted
       (da      : NSAP_addr_type;
        qos     : quality_of_service_type) : boolean_type;
primitive;

function get_sn_qos
       (subnet_id : subnet_id_type) : SN_QOS_type;
primitive;

function get_qos : quality_of_service_type;
primitive;
```

```
function make_buffer
      (data      : data_type) ; buffer_type;
primitive;

procedure merge_seg
      (buf       : buffer_type;
       so        : integer_type;
       data      : data_type);
primitive;

function min
      (i         : integer_type;
       j         : integer_type) : integer_type;
primitive;

function NPAI_addr_local
      (addr      : NPAI_addr_type) : boolean_type;
primitive;

function route
      (da        : NPAI_addr_type;
       datalen : integer_type9; sn_route_type;
primitive;

function select_segment_size
      (max_size : integer_type); integer_type;
       ipdu      : ipdu_type); integer_type;
(/ This function determines the size of the segment to
   be formed during IPDU segmentation. The maximum
   applicable size is passed as a parameter. The
   function may use additional information (such as
   knowledge of the possible IPDU route) to select a
   lower value, for example to optimise segmentation
   throughout the network /)

function size
      (data      : data_type;) : integer_type;
primitive;

function size_buf
      (buf       :buffer_type) : integer_type;
primitive;

function sn_data_maxsize
      (subnet_id : subnet_id_type) : integer_type;
primitive;


tran from INITIAL to CLOSED
when N.UNITDATA_request
provided not NSAP_addr_local (NS_Destination_Address)
```

```
begin
  nsdu.da        := NS_Destination_Address;
  nsdu.sa        := NS_Source_Address;
  nsdu.qos       := NS_Quality_of_Service;
  nsdu.data      := NS_Userdata;
  ipdu.nlpi      := CL_IP_nlpi
  ipdu.vp_id     := CL_IP_version1;
  ipdu.lifetime  := get_lifetime (nsdu.da, nsdu.qos);
  ipdu.sp        := get_seg_permitted (nsdu.da, nsdu.qos);
  ipdu.ms        := FALSE;
  ipdu.ipdu.tp   := DT;
  ipdu.da_len    := get_NPAI_len(nsdu.da);
  ipdu.da        := get_NPAI(nsdu.da);
  ipdu.sa_len    := get_NPAI_len(nsdu.sa);
  ipdu.sa        := get_NPAI(nsdu.sa);
  ipdu.data      := nsdu.data;

  ipdu.hli       := get_header_len(ipdu.da_len,
                                   ipdu.sa_len,
                                   ipdu.sp,
                                   ipdu.options);

  ipdu.seg_len  := ipdu.hli + size(ipdu.data)
  if (ipdu.sp) then
       begin
         ipdu.du_id     := get_data_unit_identifier;
         ipdu.so        := ZERO
         ipdu.tot_len   := ipdu.seg_len
       end;
  send_ipdu(ipdu);
end;

tran from INITIAL to CLOSED
when N.UNITDATA request
provided NSAP_addr_local (NS_Destination_Address)

begin
  nsdu.da    := NS_Destination_Address;
  nsdu.sa    := NS_Source_Address;
  nsdu.qos   := NS_Quality_of_Service;
  nsdu.data  := NS_Userdata;

  out N.UNITDATA_indication
     (nsdu.da, nsdu.sa, nsdu.qos, nsdu.data);
end;

tran from INITIAL to CLOSED
when SN.UNITDATA_indication
provided SN_Userdata.ipdu_tp = DT                     and
         NPAI_addr_local (SN_Userdata.da)             and
         SN_Userdata.so        = ZERO                 and
         not SN_Userdata.ms
```

```
begin
  ipdu   := SN_Userdata;
  out       N.UNITDATA_indication
           (get_NSAP_addr (ipdu.da_len, ipdu.da),
            get_NSAP_addr (ipdu.sa_len, ipdu.sa),
            get_qos (ipdu.options),
            ipdu.data);
end;

tran from INITIAL to REASSEMBLING
when SN.UNITDATA_indication
provided SN_Userdata.ipdu_tp = DT                and
            NPAI_addr_local (SN_Userdata.da)    and
            ((SN_Userdata.so > ZERO) or (SN_Userdata.ms))

begin
  ipdu := SN_Userdata;
  allocate_reassembly_resources (ipdu.tot_len);
  merge_seg
    (rcv_buf,
     ipdu.so,
     ipdu.data);

  out S.TIMER_request
    (ipdu.lifetime,
     lifetime_timer,
     null);

end;

tran from INITIAL to CLOSED
when SN.UNITDATA_indication
provided not NPAI_addr_local (SN_Userdata.da)

begin
  ipdu := SN_Userdata;

  decrement_lifetime (ipdu.lifetime);

  if (ipdu.lifetime > ZERO then
      send_ipdu (ipdu);

  else
      release_ipdu (LIFETIME_EXPIRED, ipdu);
end;


tran from REASSEMBLING to REASSEMBLING
when SN.UNITDATA_indication
provided SN_Userdata.ipdu_tp = DT          and
         SN_Userdata.du_id    = ipqu.du_id   and
         SN_Userdata.da_len   = ipdu.da_len  and
         SN_Userdata.sa       = ipdu.sa
```

```
begin
  merge_seq
    (rcv_buf,
     SN_Userdata.so,
     SN_Userdata.data);
end;

tran from REASSEMBLING to CLOSED
delay (0.0)
provided data_unit_complete (rcv_buf)

begin
  out N.UNITDATA _indication
    (get_NSAP_addr (ipdu.da_len, ipdu.da),
     get_NSAP_addr (ipdu.sa_len, ipdu.sa),
     get_qos,
     extract (rcv_buf, size_buf (rcv_buf)));

  out S.TIMER_cancel (lifetime_timer);
end;

tran from REASSEMBLING to CLOSED
when S.TIMER_indication

begin
  release_ipdu (LIFETIME_EXPIRED, ipdu);
end;
```

APPENDIX A

APPENDIX A

ERROR RECOVERY FUNCTION

A.1 INTRODUCTION

A possible need has been identified for an Error Reporting Function. The exact need, nature and use of such a function remains under study. Nevertheless, in order to allow interim use of such a function this Appendix defines an IPDU format and the circumstances in which it may be used.

NOTE

It is likely that future versions of this Standard will contain, within the body of the Standard, a description of a mechanism for an Error Reporting Function. It is likely that this will be incompatible with the mechanism described in this Appendix.

This Appendix does not form part of the Standard.

A.2 USE OF ERROR REPORTING FUNCTION

This function may return an Error IPDU to the source network entity whenever an IPDU is discarded. The situations in which an IPDU may be discarded, and an Error IPDU may be issued, are as follows:

- the lifetime of an IPDU has expired;
- the destination is unreachable;
- the operation of congestion control requires that the IPDU be discarded;
- an unsupported or unrecognized option appears in the IPDU;
- the IPDU Header Error Detection function has detected an error;
- a violation of protocol procedure has occurred.

The Error IPDU identifies the discarded data unit, specifies the type of error detected, and gives the location where the error was detected. Part or all of the discarded data unit may be included as part of the error report data field. Error reports are not necessarily generated in all of the cases described above, by all network entities. Error reports are not sent to report the loss of an Error IPDU. Non receipt of an Error IPDU does not imply correct delivery of a Data IPDU.

This Appendix does not describe the procedure to be followed by a network entity upon receipt of an Error IPDU.

A.3 ERROR REPORT IPDU

bit 8 — bit 1	Octet
Network Layer Protocol Id.	1
Length Indicator	2
Version/Protocol ID	3
Lifetime	4
SP MS Type	5
Segment Length	6 7
Reserved for use by IPDU Header Error Detection Function	8 9
Destination Address Length Indicator	10
Destination Address	11 m-1
Source Address Length Indicator	m
Source Address	m+1 n-1
Data Unit Identifier	n n+1
Segment Offset	n+2 n+3
Total Length	n+4 n+5
Options	n+6 p
Error Report Data Field	p+1 z

Figure A.1 Error Report IPDU

A.3.1 Fixed Fields

- Network Layer Protocol Id. see 2.4.2.1
- Length Indicator see 2.4.3.2
- Version/Protocol ID see 2.4.3.3
- Lifetime see 2.4.3.4
- SP, MS see 2.4.3.5
- Type see 2.4.3.6
- Segment Length see 2.4.3.7
- IPDU Error Detection Function see 2.4.3.8

A.3.2 Addresses

See 2.4.4. The Destination Address specifies the original source of the IPDU discarded. The Source Address specifies the intermediate system or end system network Entity initiating the Error Report IPDU.

A.3.3 Options

See 2.4.5

A.3.4 Reason for Discard

This paramater is only valid for the Error Report IPDU. It provides a report on the discarded IPDU.

- Parameter code : 1100 0001
- Parameter length : one octet
- Parameter value : the following values, in binary, specify the type of error:

  1: incorrect source routing or Destination Address unreachable;
  2: IPDU Header Error detection
  3: subject IPDU discarded due to lifetime expiration;
  4: subject IPDU discarded due to presence of unsupported options;
  5: subject IPDU discarded due to congestion;
  6: any other protocol procedure violation.

A.3.5 Error Report Data Field

This field provides all or a portion of the discarded IPDU. The octets comprising this field contain the rejected or discarded IPDU up to and including the octet which caused the rejection/discard.