iNA960 ARCHITECTURE

REFERENCE MANUAL

Manual Order Number: 122194-001

Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Intel retains the right to make changes to these specifications at any time, without notice. Contact your local sales office to obtain the latest specifications before placing your order.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may only be used to identify Intel products:

BITBUS	i_m	iRMX	Plug-A-Bubble
COMMputer	iMMX	iSBC	PROMPT
CREDIT	Insite	iSBX	Promware
Data Pipeline	$int_e l$	iSDM	QueX
Genius	int_eIBOS	iSXM	QUEST
i	Intelevision	Library Manager	Ripplemode
$\hat{i}$	int_eligent Identifier	MCS	RMX/80
I^2ICE	int_eligent Programming	Megachassis	RUPI
ICE	Intellec	MICROMAINFRAME	Seamless
iCS	Intellink	MULTIBUS	SOLO
iDBP	iOSP	MULTICHANNEL	SYSTEM 2000
iDIS	iPDS	MULTIMODULE	UPI
iLBX			

REV.	REVISION HISTORY	DATE	APPD.
-001	Original issue		

PREFACE

This manual provides all of the architecture details necessary to use iNA 960 Release 1. This manual is intended for use by system designers and application programmers.

This manual contains six chapters:

Chapter 1, "Introduction" describes the relationship of the iNA 960 architecture to the ISO model and presents an overview of each layer incorporated into iNA 960.

Chapter 2, "Data Link" describes the architecture of the data link layer.

Chapter 3, "Network Layer" describes the minimal network protocol in iNA 960 release 1.

Chapter 4, "Transport Layer (Connection Oriented)" describes the connection oriented transport layer protocol for iNA 960 release 1.

Chapter 5, "Transport Layer (Connectionless)" describes the connectionless transmission of of data and protocol for iNA 960 release 1.

Chapter 6, "Network Management" describes the planning, operation and maintenance facilities in iNA 960 release 1.

RELATED PUBLICATIONS

For further information on iNA 960 Release 1, refer to the following publication:

- iNA 960 Programmers Reference Manual, 122193

Notational Conventions

UPPERCASE	Characters shown in uppercase must be entered in the order shown. You may enter the characters in uppercase or lowercase.
italic	Italic indicates a meta symbol that may be replaced with an item that fulfills the rules for that symbol. The actual symbol may be any of the following:
directory-name	Is that portion of a *pathname* that acts as a file locator by identifying the device and/or directory containing the *filename*.
filename	Is a valid name for the part of a *pathname* that names a file.
pathname	Is a valid designation for a file; in its entirety, it consists of a *directory* and a *filename*.
pathname1, pathname2, ...	Are generic labels placed on sample listings where one or more user-specified pathnames would actually be printed.
system-id	Is a generic label placed on sample listings where an operating system-dependent name would actually be printed.
V*x.y*	Is a generic label placed on sample listings where the version number of the product that produced the listing would actually be printed.
[ ]	Brackets indicate optional arguments or parameters.
{ }	One and only one of the enclosed entries must be selected unless the field is also surrounded by brackets, in which case it is optional.
{ }...	At least one of the enclosed items must be selected unless the field is also surrounded by brackets, in which case it is optional. The items may be used in any order unless otherwise noted.
|	The vertical bar separates options within brackets [ ] or braces { }.
...	Ellipses indicate that the preceding argument or parameter may be repeated.
[,...]	The preceding item may be repeated, but each repetition must be separated by a comma.
punctuation	Punctuation other than ellipses, braces, and brackets must be entered as shown. For example, the punctuation shown in the following command must be entered: `SUBMIT PLM86(PROGA,SRC,'9 SEPT 81')`
user input	In interactive examples, user input lines are printed in white on black to differentiate them from system output.
<cr>	Indicates a carriage return.

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

TABLES

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 RELATIONSHIP OF THE ARCHITECTURE OF iNA 960 TO THE ISO MODEL

Figure 1-1 shows the layered structure of the ISO model upon which the architecture of iNA 960 is based. Of the several layers in the model, only the following layers are implemented in the iNA 960:

Data Link
Network
Transport
Network Management

These are the layers the architectures of which are discussed in the remaining sections of this manual.

1.1.1 Layers Not Incorporated in iNA 960

The physical layer provides the (logical) functional and procedural characteristics to enable nodes to activate, maintain and deactivate the physical link through the transmission media (it is not itself a part of "the physical media"). It deals only with bits passed to and from the Data Link Layer and because of this association is of interest. The physical layer is present in the hardware supported by the iNA960, but it is not a part of the iNA 960 architecture. Therefore, its architecture is not discussed in this manual.

Similarly, in so far as the Session Layer's interface with the transport layer affects the architecture of the Transport Layer, there is some mention of some of its characteristics, but its architecture is not discussed.

All other layers are not represented at all (except in figures like Figure 1-1) in this manual.

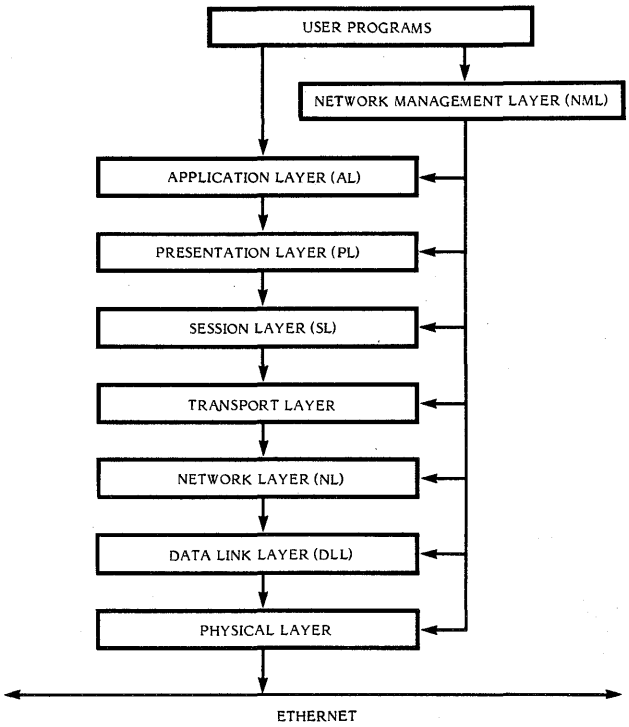


Figure 1-1. The Layered Network Architecture

1.1.2 The Layers in iNA 960

The Data Link Layer is responsible for transfer of information over the physical transmission media. The Data Link Layer synchronizes information to delimit the flow of bits to and from the Physical Layer and give it identity. Also, it includes means for detecting transmission errors.

If this layer is implemented with a fully functional protocol, higher level layers (Network, Transport, etc.) can assume that received information is free of transmission errors (although there is always a finite probability that there will be undetected errors).

The architecture of the Data Link Layer is discussed in Chapter 2 of this manual.

The Network Layer is responsible for the routing and relay functions through switched telecommunications media. It provides network connection between transport entities and can multiplex two or more network connections over a simple data link access to a relay mode.

The architecture of the Network Layer is described and discussed in Chapter 3 of this manual.

The Transport Layer is the lowest layer whose peer entities are always in the communicating end systems. The functions of this Layer are not involved in intervening telecommunications network (or relay) nodes. The basic purpose of the Transport Layer is to provide a consistent transport service in association with the lower three layers (Network, Data Link and Physical). It optimizes the use of the network services and corrects for deficiencies in quality of service to meet the requirements of upper layers and communicating application processes.

Among the functions that can be performed by the Transport Layer are establishment of transport connections, error recovery, multiplexing, flow control and error detection. As with other layers, only the functions that are needed for the particular situation are involved for a specific communication. These can be negotiated during establishment of the communication.

Connection oriented mode Transport Layer architecture is described and discussed in Chapter 4 of this manual. Connectionless mode Transport Layer architecture is discussed in Chapter 5.

The Network Management Layer provides planning, operation and maintenance facilities to the network. It gathers network usage information for the user, provides initialization, termination monitoring and performance optimization. It deals with and manages detection, isolation, amputation and repair of network faults.

The architecture of Network Management Layer is described and discussed in Chapter 6 of this manual.

CHAPTER 2
DATA LINK

2.1 INTRODUCTION

This chapter is based on the standard for the data link layer as defined in IEEE Project 802 Local Area Network Standard Document P802.2 (Logical Link Control) and P802.3 (CSMA/C Access Method and Physical Layer Specifications). The standard was produced to facilitate the interconnection of computers and terminals on a Local Area Network (LAN). It is related to the other standards by the Reference Model for Open Systems Interconnection.

As defined in these documents, the Data Link Layer is divided into two sublayers; the Logical Link Control (LLC) and the Memory Access Control (MAC) sublayers. The LLC sublayer (see Figure 2-1) is the top sublayer in the data link layer. Separate standards describe each medium access method individually and indicate the additional features and functions that are provided by the Medium Access Control (MAC) sublayer in each case to complete the functionality of the Data Link Layer as defined in the IEEE 802 architectural reference model.

This chapter first describes the LLC Sublayer Interface Service Specifications to the Network Layer (Layer 3), to the MAC sublayer, and to the LLC Sublayer Management function. Description of the specified interface service to the Network Layer provides a description of the various services that the Logical Link Control sublayer, plus underlying layers and sublayer, offer to the Network Layer, as viewed from the Network Layer. Description of the specified interface service to the MAC sublayer provides a description of the services that the LLC sublayer requires of the MAC sublayer. These services are defined in such a way as to be independent of the form of the medium access methodology, and of the nature of the medium itself. Description of the specified interface service to the LLC Sublayer Management function provides a description of the management services that are provided to the LLC sublayer. All of the above Interface Service Specifications are given in the

form of primitives that represent in an abstract way the logical exchange of information and control between the LLC sublayer and the identified service function (Network Layer, MAC sublayer or LLC Sublayer Management function). They do not specify or constrain the implementation of entities or interfaces.

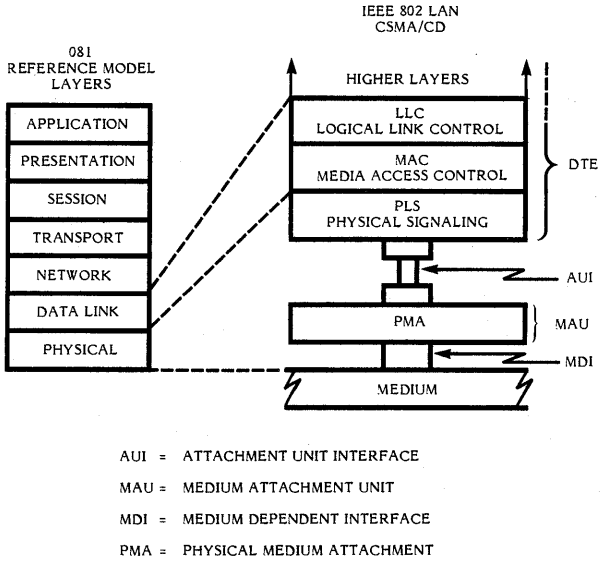


Figure 2-1. Relationship to LAN Reference Model

This chapter provides a description of the peer-to-peer protocol procedures that are defined for the transfer of information and control between any pair of Data Link Layer service access points on a local area network. The LLC Procedures are independent of the type of medium access method used in the particular local area network.

To satisfy a broad range of potential applications, two types of data link control operations are included in IEEE 802.2, but only one of these is of interest and discussed in this manual (see section 2.5). It (see section 2.7) provides a data-link-connectionless service across a data link with minimum protocol com-

plexity. This type of operation may be useful when higher layers provide any essential recovery and sequencing services so that these do not need replicating in the Data Link Layer. In addition, this type of operation may prove useful in applications where it is not essential to guarantee the delivery of every Data Link Layer data unit. This type of service is described in this chapter in terms of "logical data links".

Two distinct "classes" of LLC operation are identified. Class I provides data-link-connectionless service only. Class II provides data-link-connection-oriented service plus data-link-connectionless service. Either class of operation may be supported.

The basic protocols described herein are peer protocols for use in multi-station, multi-access environments. Because of the multi-station, multi-access environment, it must be possible for a station to be involved in a multiplicity of peer protocol data exchanges with a multiplicity of different stations over a multiplicity of different logical data links and/or data link connections that are carried by a single Physical Layer over a single physical medium. Each unique to-from pairing at the Data Link Layer must define a separate logical data link or data link connection with separate logical parameters and variables. Except where noted, the procedures described in this chapter relate to each Data Link Layer logical data link or data link connection separately and independently from any other logical data link or data link connection that might exist at the stations involved.

2.1.1 Standards Compatibility

The peer protocol procedures defined in section 2.6 use some of the concepts and principles, as well as commands and responses, of the balanced link control procedures known as Asynchronous Balanced Mode (ABM), as defined in ISO 6256-1979 and ANSI X3.66-1979. (The ABM procedures provided the basis upon which the CCITT Recommendation X.25 Level 2 LAPB procedures were defined.) The frame structure defined for the Data Link Layer procedures as a whole is defined in part in sections 2.4 through 2.7 and in part in those standards that define the various Medium Access Control (MAC) procedures in sections 2.10 through 2.12 . The combination of a MAC sublayer address and an LLC sublayer address is unique to each Data Link Layer service access point in the local area network.

2.1.2 References

- ISO

  ISO/DIS 4335 Revised, Data Communication - High-level data link control procedures - Consolidation of elements of procedures

  ISO/DIS 7809, Data Communication - High-level data link control procedures - Consolidation of classes of procedures

  ISO/DIS 7498, Data Processing - Open Systems Interconnection, Basic Reference Model, February 4, 1982.

- CCITT

  Recommendation X.25, Interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) for terminals operating in the packet mode on public data networks.

  CCITT COMVII R9(C), Reference Model on Open Systems Interconnection for CCITT Applications, March 1982.

- ANSI

  X3.66-1979, American National Standard for Advanced Data Communications Control Procedures (ADCCP).

2.1.3 Definitions

NOTE

For definitions of abbreviations used in the following, see section 2.1.4.

In this chapter, the following definitions apply:

Accept: The condition assumed by a LLC upon accepting a correctly received PDU for processing.

Address Fields (DSAP and SSAP): The ordered pair of service access point addresses at the beginning of a LLC PDU which identifies the LLC(s) designated to receive the PDU and the LLC sending the PDU. Each address field is one octet in length.

Basic Status: A LLC's capability to send or receive a PDU containing an information field.

Command: In data communication, an instruction represented in the control field of a PDU and transmitted by a LLC. It causes the addressed LLC(s) to execute a specific data link control function.

Command PDU: All PDUs transmitted by a LLC in which the C/R bit is equal to 0.

Control Field (C): The octet immediately following the DSAP and SSAP address fields of a PDU. The content of the control field is interpreted by the receiving destination LLC(s) designated by the DSAP address field:

- As a command, from the source LLC designated by the SSAP address field, instructing the performance of some specific function;

- As a response, from the source LLC designated by the SSAP address field.

Data Link: An assembly of two or more terminal installations and the interconnecting communications channel operating according to a particular method that permits information to be exchanged; in this context, the term "terminal installation" does not include the data source and the data sink.

Data Link Layer: The conceptual layer of control or processing logic existing in the hierarchical structure of a station that is responsible for maintaining control of the data link. The data link layer functions provide an interface between the station higher layer logic and the data link. These functions include address/control field

interpretation, channel access and command PDU/response PDU generation, transmission and interpretation.

Exception Condition: The condition assumed by a LLC upon receipt of a command PDU which it cannot execute due to either a transmission error or an internal processing malfunction.

Global (Broadcast) DSAP Address: The predefined LLC DSAP address (all ones) used as a broadcast (all parties) address. It can never be the address of a single LLC on the data link.

Group (Multicast) DSAP Address: A destination address assigned to a collection of LLCs to facilitate their being addressed collectively. The least significant bit must be set equal to "1".

Higher Layer: The conceptual layer of control or processing logic existing in the hierarchical structure of a station that is above the data link layer and upon which the performance of data link layer functions depend; for example, device control, buffer allocation, LLC station management, etc.

Information Field (I): The sequence of octets occurring between the control field and the end of the PDU. The information field contents of I, TEST and UI PDUs are not interpreted at the LLC sublayer.

Invalid Frame: A PDU that: a) Does not contain an integral number of octets, b) Does not contain at least two address octets and a control octet, or c) Is identified by the Physical Layer or MAC Sublayer as containing data bit errors.

Protocol Data Unit (PDU): The sequence of contiguous octets delivered as a unit from or to the MAC Sublayer. A valid PDU is at least three octets in length and contains two address fields, and a control field. A PDU may or may not include an information field in addition.

Response: In data communications, a reply represented in the control field of a response PDU. It advises the addressed destination LLC with respect to the action taken by the source LLC to one or more command PDUs.

Response PDU: All PDUs sent by a LLC in which the C/R bit in the SSAP is equal to "1".

LLC: That part of a data station that supports the Logical Link Control functions of one or more logical links. The LLC generates command PDUs and response PDUs for transmission, and interprets received command PDUs and response PDUs. Specific responsibilities assigned to a LLC include:

- Initiation of control signal interchange;
- Organization of data flow;
- Interpretation of received command PDUs and generation of appropriate response PDUs, and;
- Actions regarding error control and error recovery functions in the LLC Sublayer.

2.1.4 Abbreviations

The following abbreviations are used in this chapter:

ABM	Asynchronous Balanced Mode
ACK	ACKnowledge
ADCCP	Advanced Data Communication Control Procedures
ADM	Asynchronous Disconnected Mode
ANSI	American National Standards Institute
C	Command
CAI	Connectionless Acknowledged Information
CCITT	International Telegraph and Telephone Consultative Committee
COMVII	Commission VII
C/R	Command/Response
DA	Destination Address
DCE	Data Circuit-terminating Equipment

DIS	Draft International Standard
DISC	DISConnect
DM	Disconnected Mode
DSAP	Destination Service Access Point
DTE	Data Terminal Equipment
F	Final
FCS	Frame Check Sequence
FRMR	FRaMe Reject
HDLC	High level Data Link Control
I	Information
I	Information transfer format
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Organization for Standardization
LAN	Local Area Network
LAPB	Link Access Procedure, Balanced
LLC	Logical Link Control
LSAP	Link layer Service Access Point
LSB	Least Significant Bit
LSDU	Link layer Service Data Unit
M	Modifier function bit
MAC	Medium Access Control
MUI	Mode-independent Unnumbered Information
N(R)	Receive sequence Number
N(S)	Send sequence Number
OSI	Open System Interconnection
P	Poll
PDU	Protocol Data Unit
P/F	Poll/Final
PHY	PHYsical
R	Response
REJ	REJect
RNR	Receive Not Ready
RR	Receive Ready
S	Supervisory format
S	Supervisory function bit

SA	Source Address
SABM	Set Asynchronous Balanced Mode
SAP	Service Access Point
SSAP	Source Service Access Point
TEST	TEST
U	Unnumbered format
UA	Unnumbered Acknowledgement
V(R)	Receive state Variable
V(S)	Send state Variable
XID	eXchange IDentification

2.2 LLC SUBLAYER INTERFACE SERVICE SPECIFICATIONS (with other than LLC sublayer management function)

This and following sections cover the services required of or by the Logical Link Control (LLC) sublayer at the logical interfaces with the Network Layer, the MAC sublayer, and the LLC Sublayer Management function.

In general, the services of a layer (or sublayer) are the capabilities which it offers to a user in the next higher layer (or sublayer). In order to provide its service, a layer (or sublayer) builds its functions on the services which it requires from the next lower layer (or sublayer). Figure 2-2 illustrates this notion of service hierarchy and shows the relationship of the two correspondent n-users and their associated n-layer (or sublayer) peer protocol entities.

Services are specified by describing the information flow at the interface between the n-user and the n-layer (or sublayer). This information flow is modeled by discrete, instantaneous interface events, which characterize the provision of a service. Each event consists of passing a service primitive from one layer (or sublayer) to the other through an n-layer (or sublayer) service access point associated with an n-user. Service primitives convey the information required in providing a particular service. These service primitives are an abstraction in that they specify only the service provided rather than the means by which the service is provided. This definition of service is independent of any particular interface implementation

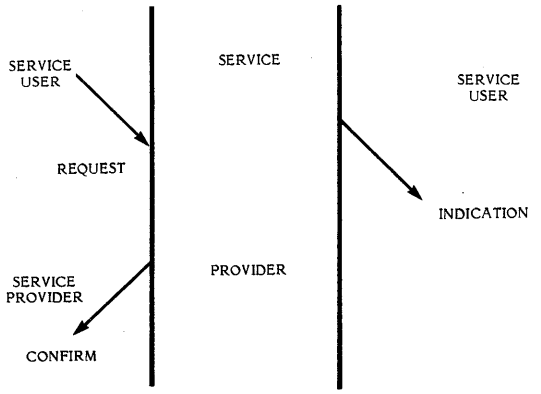


Figure 2-2. Service Hierarchy

Services are specified by describing the service primitives and parameters which characterize each service. A service may have one or more related primitives which constitute the interface activity that is related to the particular service. Each service primitive may have zero or more parameters which convey the information required to provide the service.

Primitives are of three generic types:

REQUEST — The request primitive is passed from the n-user to the n-layer (or sublayer) to request that a service be initiated.

INDICATION — The indication primitive is passed from the n-layer (or sublayer) to the n-user to indicate an internal n-layer (or sublayer) event which is significant to the n-user. This event may be logically related to a remote service request, or may be caused by an event internal to the n-layer (or sublayer).

CONFIRM — The confirm primitive is passed from the n-layer (or sublayer) to the n-user to convey the results of the associated previous service request. This primitive may indicate either failure to comply or

some level of compliance. It does not necessarily indicate any activity at the remote peer interface.

Possible relationships among primitive types are illustrated by the time sequence diagrams shown in Figure 2-3. The figure also indicates the logical relationship of the primitive types. Primitive types which occur earlier in time and are connected by dotted lines in the diagrams are the logical antecedents of subsequent primitive types. Note that the logical and time relationship of the indication and the response primitive types are specified by the semantics of a particular service.

2.2.1 Network Layer/LLC Sublayer Interface Service Specifications

The following specifies the services required of the Logical Link Control (LLC) sublayer by the Network Layer, as viewed from the Network Layer, to allow a local Network Layer entity to exchange packets with remote peer Network Layer entities. The services are described in an abstract way and do not imply any particular implementation or any exposed interface.

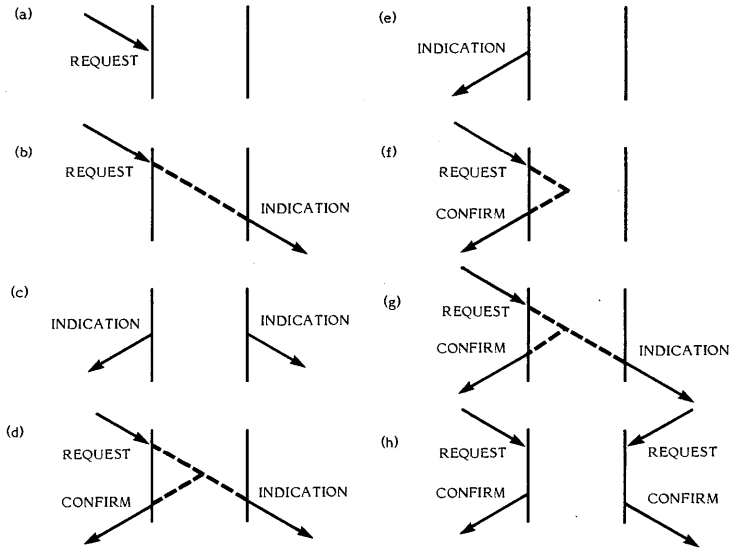


Figure 2-3. Time Sequence Diagrams

Only unacknowledged connectionless service is supported in iNA960 (R1). Unacknowledged connectionless data transfer service provides the means by which network entities can exchange link service data units (LSDUs) without the establishment of a data link level connection. The data transfer can be point-to-point, multi-cast, or broadcast.

The primitives associated with unacknowledged connectionless data transfer are:

L_DATA.request
L_DATA.indication

The L_DATA.request primitive is passed to the LLC sublayer to request that an LSDU be sent using unacknowledged connectionless procedures. The L_DATA.indication primitive is passed from the LLC sublayer to indicate the arrival of an LSDU.

2.2.2 Detailed Service Specifications

This section describes in detail the primitives and parameters associated with the identified services. Note that the parameters are specified in an abstract sense. The parameters specify the information that must be available to the receiving entity. A specific implementation is not constrained in the method of making this information available. For example, the l_sdu parameter associated with some of the data transfer service primitives may be provided by actually passing the link service data unit, by passing a pointer, or by other means. Similarly, an implementation of connection-oriented services may make use of a locally significant connection identifier to imply local and remote address parameters. The LLC sublayer may also provide local response mechanisms for request type primitives:

L_DATA.request

Function: This primitive is the service request primitive for the unacknowledged connectionless data transfer service.

Semantics of the Service Primitive: The primitive must provide parameters as follows:

L_DATA.request (
local_address,
remote_address,
l_sdu,
quality
)

The local_address and remote_address parameters specify the local and remote LSAPs involved in the transfer by providing, at a minimum, the logical concatenation of the MAC address field (SA and/or DA) and the LLC address field (SSAP and/or DSAP). The remote_address may specify either an individual or a group address. The l_sdu parameter specifies the link service data unit to be transferred by the link layer entity. The quality parameter specifies the service class desired for the data unit transfer.

When Generated: This primitive is passed from the network layer to the LLC sublayer to request that a LSDU be sent to one or more remote LSAP(s) using unacknowledged connectionless procedures.

Effect on Receipt: Receipt of this primitive causes the LLC sublayer to attempt to send the LSDU using unacknowledged connectionless procedures.

Additional Comments: This primitive is independent of any connection with the remote LSAP.

A possible logical sequence of primitives associated with successful unacknowledged connectionless data transfer is illustrated in Figure 2-3(c).

L_DATA.indication

Function: This primitive is the service indication primitive for the unacknowledged connectionless data transfer service.

Semantics of the Service Primitive: The primitive must provide parameters as follows:

```
L_DATA.indication (
                    local_address,
                    remote_address,
                    l_sdu
                    quality
                    )
```

The local_address and remote_address parameters specify the local and remote LSAPs involved in the transfer. The local address may be the address of a local LSAP, or may be a group address specifying multiple LSAPs, including a local LSAP. The l_sdu parameter specifies the link service data unit which has been received by the LLC sublayer entity. The quality parameter specifies the service class desired for the data unit transfer.

When Generated: This primitive is passed from the LLC sublayer to the network layer to indicate the arrival of an LSDU from the specified remote entity.

Effect on Receipt: The effect of receipt of this primitive by the network layer is unspecified.

Additional Comments: This primitive is independent of any connection with the remote LSAP.

In the absence of errors, the contents of the l_sdu parameter are logically complete and unchanged relative to the l_sdu parameter in the associated L_DATA.request primitive.

2.2.3 LLC Sublayer/MAC Sublayer Interface Service Specification

The following specifies the services required of the Medium Access Control (MAC) sublayer by the Logical Link Control (LLC) sublayer to allow the local LLC sublayer entity to exchange LLC data units with peer LLC sublayer entities. The services are described in an abstract way and do not imply any particular implementation or any exposed interface. The following are the primitives involved:

- MA_DATA.request
- MA_DATA.indication
- MA_DATA.confirm

The following provide detailed service specification:

MA_DATA.request

Function: This primitive defines the transfer of a MSDU from a local LLC sublayer entity to a single peer LLC entity, or multiple peer LLC entities in the case of group addresses.

Semantics of the Service Primitive: The semantics of the primitive are as follows:

```
MA_DATA.request (
                    destination_address,
                    m_sdu,
                    requested_quality
                    )
```

The destination_address parameter must specify either an individual or a group MAC entity address. It must contain sufficient information to create the DA field that is appended to the frame by the local MAC sublayer entity as well as any lower level address information. The m_sdu parameter specifies the MAC service data unit to be transmitted by the MAC sublayer entity, which includes the DSAP, SSAP, C, and information (if present) fields as specified in section 2.4, as well as sufficient information for the MAC sublayer entity to determine the length of the data unit. The requested_quality parameter specifies the service class desired for the data unit transfer.

When Generated: This primitive is generated by the LLC sublayer entity whenever a MSDU must be transferred to a peer LLC entity or entities. This can be as a result of a request from higher layers or protocol or from a MSDU generated internally to the LLC sublayer, such as required by Type 2 operation.

Effect of Receipt: The receipt of this primitive must cause the MAC entity to append all MAC specified fields, including DA, SA, and any fields that are unique to the particular medium access method, and pass the properly formatted frame to the lower layers of protocol for transfer to the peer MAC sublayer entity or entities.

Additional Comments: None.

MA_DATA.indication

Function: This primitive defines the transfer of a MSDU from the MAC sublayer entity to the LLC sublayer entity or entities in the case of group addresses. In the absence of errors, the contents of the m_sdu parameter are logically complete and unchanged relative to the m_sdu parameter in the associated MA_DATA.request.

Semantics of the Service Primitive: The semantics of the primitive are as follows:

```
MA_DATA.indication (
                    destination_address,
                    source_address,
                    m_sdu,
                    reception_status
                    requested_quality
                    )
```

The destination_address parameter must be either an individual or a group address as specified by the DA field of the incoming frame. The source_address parameter must be an individual address as specified by the SA field of the incoming frame. The m_sdu parameter specifies the MAC service data unit as received by the local MAC entity. The reception_status parameter indicates the success or failure of the incoming frame. The requested quality parameter specifies the service class desired for this data unit transfer.

When Generated: The MA_DATA.indication is passed from the MAC sublayer entity to the LLC sublayer entity or entities to indicate the arrival of a frame at the local

MAC sublayer entity. Such frames are reported only if they are validly formatted, received without error, and their destination address designates the local MAC entity.

Effect of Receipt: The effect of receipt of this primitive by the LLC sublayer is unspecified.

Additional Comments: If the local MAC sublayer entity is designated by the destination address parameter of an MA_DATA.request, the indication primitive will also be invoked by the MAC entity to the local LLC entity. This full duplex characteristic of the MAC sublayer may be due to unique functionality within the MAC sublayer or full duplex characteristics of the lower layers, (e.g., all frames transmitted to the broadcast address will invoke MA_DATA.indications at all stations in the network including the station that generated the request).

MA_DATA.confirm

Function: This primitive has local significance and must provide an appropriate reply to the LLC sublayer MA_DATA.request primitive signifying the success or failure of the request.

Semantics of the Service Primitive: The semantics of this primitive are as follows:

```
MA_DATA.confirm (
                    transmission_status
                    provided_quality
                    )
```

The transmission_status parameter is used to pass status information back to the local requesting LLC sublayer entity. It is used to indicate the success or failure of the previous associated MA_DATA.request. The types of failures that can be associated with this primitive depend on the particular implementation as well as the type of Medium Access Control sublayer that is used, (e.g., excessive collisions may be a failure returned by a CSMA/CD MAC sublayer entity). The provided_qual-

ity parameter specifies the service class that was provided for the data unit transfer.

When Generated: This primitive is generated in reply to an MA_DATA.request from the local LLC sublayer entity.

Effect of Receipt: The effect of receipt of this primitive by the LLC sublayer is unspecified.

Additional Comments: It is assumed that sufficient information is available to the LLC sublayer to associate the confirm with the appropriate request.

2.3 LLC SUBLAYER/LLC SUBLAYER MANAGEMENT FUNCTION INTERFACE SERVICE SPECIFICATION

(This matter is the subject of further on-going study and resolution.)

2.4 LLC PROTOCOL DATA UNIT (PDU) STRUCTURE

The following defines in detail the Logical Link Control (LLC) Protocol Data Unit (PDU) structure for data communication systems using bit-oriented procedures. It defines the relative positions of the various components of the Protocol Data Unit (PDU). It defines the method for representing Data Link Layer service access point addresses (to or from Network Layer entities). It defines a partition of these addresses into individual and group addresses. Details of the control and information field allocation are specified in section 2.6.

2.4.1 LLC PDU Format

All LLC PDUs must conform to the format shown in Figure 2-4.

2.4.2 Elements of the LLC PDU

Address Fields: Each LLC PDU must contain two address fields: the Destination Service Access Point (DSAP) address field and the Service Access Point (SSAP)

Address field, in that order. Each address field must contain only a single address. The DSAP Address field must identify the one or more service access points for which the LLC information field is intended. The SSAP Address field must identify the specific service access point from which the LLC information field was initiated.

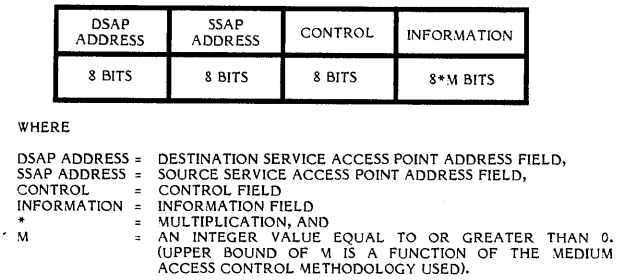


Figure 2-4. LLC PDU Format

Address Representation: The representation of each address field must be as shown in Figures 2-5(a) and 2-5(b).

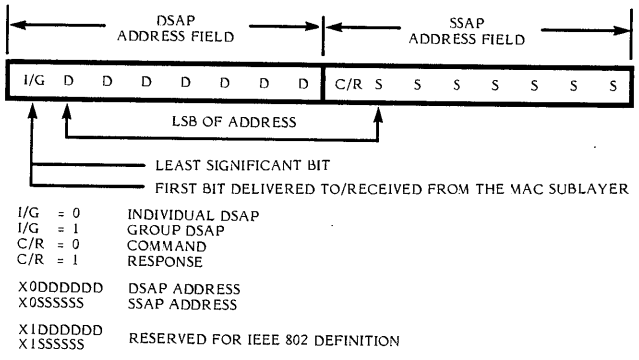


Figure 2-5(a). DSAP and SSAP Address Field Formats

Each address field must contain one octet.

Each address field must contain 7 bits of actual address, and one bit that must be used in the DSAP Address field to identify the DSAP address as either an individual or a group address (called the address type designation bit) and in the SSAP Address field to identify that the LLC PDU is a command or a response (called the command/response identifier bit).

The address type designation bit must be located in the least significant bit position of the DSAP address field. If this bit is "0", it must indicate that the address in an individual DSAP address. If this bit is "1", it must indicate that the address is a group DSAP address that identifies none, one or more, or all of the service access points that are serviced by the LLC entity.

The command/response identifier bit must be located in the least significant bit position of the SSAP address field. If this bit is "0", it must indicate that the LLC PDU is a command. If this bit is "1", it must indicate that the LLC PDU is a response.

Address Usage: An individual address must be usable as both a SSAP and a DSAP address; a null address must be usable as both a SSAP and a DSAP address; a group address must be usable only as a DSAP address.

All "1"s in the DSAP address field (i.e., the address type designation bit set to "1", and the seven address bits set to "1") is predefined to be the "Global" DSAP address. This DSAP address designates a group consisting of all DSAPs actively being serviced by the underlying MAC Service Access Point Address(es).

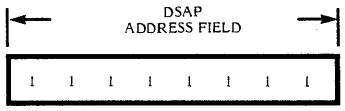


Figure 2-5(b). Global DSAP Address Field Format

All "0"s in the DSAP or SSAP address field (i.e., the address type designation bit set to "0", and the seven address bits set to "0") is predefined to be the "Null" address. The Null service access point address designates the LLC that is associated with the underlying MAC Service Access Point address, and is NOT used to identify any service access point to the Network Layer or any service access point to an associated Layer Management function.

Addresses 01000000 and 11000000 are designated as the individual and group addresses, respectively, for an LLC sublayer management function at the station. Other addresses with the next to low-order bit set to "1" are reserved for IEEE 802 definition.

Control Field: The control field must consist of one octet which must be used to designate command and response functions, and which must contain sequence numbers when required. The content of this field must be as described in section 2.6.

Information Field: The information field must consist of an integral number (including zero) of octets.

Bit Order: Addresses, commands and responses, and sequence numbers must be delivered to/received from the MAC sublayer least significant bit first (i.e., the first bit of a sequence number that is delivered/received must have the weight 2**0). The information field must be delivered to the MAC sublayer in the same bit order as received from Layer 3. The information field must be delivered to Layer 3 in the same bit order as received from the MAC sublayer.

Invalid LLC PDU: An invalid LLC PDU is defined as one which meets at least one of the following conditions:

- It is identified as such by the Physical Layer or the medium access control (MAC) sublayer.

- It is not an integral number of octets in length.

- It does not contain two properly formatted address fields, one control field, and, optionally, an information field in their proper order.

- Its length is less than 3 octets.

Invalid LLC PDUs must be ignored.

2.5 LLC TYPES AND CLASSES OF PROCEDURE

LLC defines two types of operation (Type 1 and Type 2) for data communication between service access points. Only Type 1 is of interest in this manual so Type 2 is not discussed.

With Type 1 operation, PDUs must be exchanged between LLCs without the need for the establishment of a data link connection. In the LLC sublayer, these PDUs must not be acknowledged, nor may there be any flow control or error recovery in the Type 1 procedures.

Two classes of LLC are defined. A Class I LLC supports Type 1 operation only, whereas a Class II LLC supports both Type 1 and Type 2 operations (Figure 2-6).

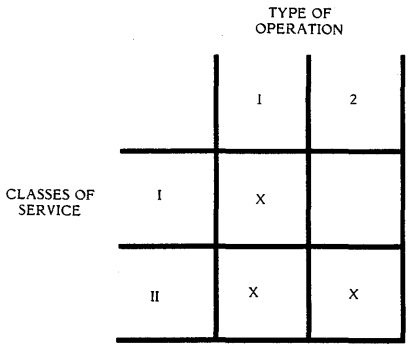


Figure 2-6. Classes of Service

This means that all LLCs on a local area network must have Type 1 operation in common. In a Class II LLC, the support of Type 1 must be totally independent of the modes or change of modes of the Type 2 operation in that same LLC.

Class I LLC: Class I LLCs must support Type 1 operation only. Class I service must be applicable to individual, group, global, and null DSAP addressing, and applications requiring no data link layer acknowledgement or flow control procedures. The set of command PDUs and response PDUs supported in Class I service are:

	Commands	Responses
Type 1:	UI	
	XID	XID
	TEST	TEST

2.6 LLC ELEMENTS OF PROCEDURE

The following specifies the elements of the local area network logical link control (LLC) procedures for code-independent data communication using the LLC PDU structure (see section 2.4).

These LLC elements of procedure are defined specifically in terms of the actions that must occur in the LLC on receipt of commands, and occasionally on receipt of a reply to a command, over a logical data link (Type 1).

2.6.1 Control Field Formats

The three formats defined for the control field (Figure 2-7) must be used to perform numbered information transfer, numbered supervisory transfer, unnumbered control and unnumbered information transfer functions. The numbered information transfer and supervisory transfer functions apply only to Type 2 operation. The unnumbered control and unnumbered information transfer functions apply either to Type 1 or Type 2 operation (but not both) depending upon the specific function selected.
Information Transfer Format - I: Not of interest in this manual.

Supervisory Format - S: Not of interest in this manual.

Unnumbered Format - U: The U-format PDUs must be used in either Type 1 or Type 2 operation, depending upon the specific function utilized, to provide additional data link control functions and to provide unsequenced information transfer. The U-format PDUs must contain no sequence numbers, but must include a P/F bit that must be set to "1" or "0".

2.6.2 Control Field Parameters

The various parameters associated with the control field formats are described in the following.

Type 1 Operation Parameters: The only parameter that exists in Type 1 operation is the Poll/Final (P/F) bit. The P/F bits set to "1" is only to be used in Type 1 operation with the XID and TEST command/response PDU functions. The Poll (P) bit set to "1" must be used to solicit (poll) a corresponding response PDU with the F bit set to 1 from the addressed LLC. The Final (F) bit set to "1" must be used to indicate that response PDU that is sent by the LLC as the result of a soliciting (poll) command PDU (P bit set to "1").

Type 2 Operation Parameters: Not of interest in this manual.

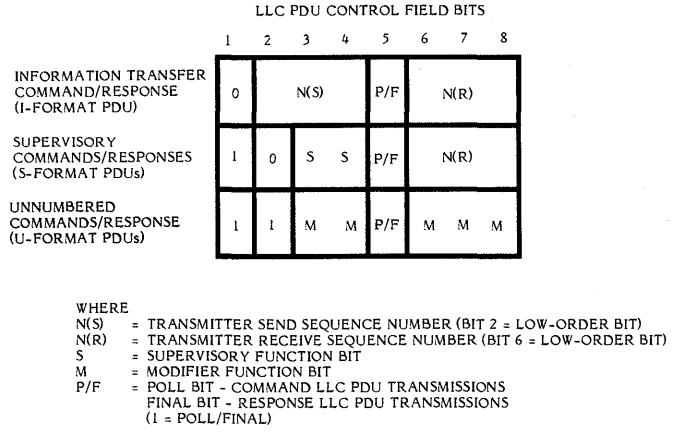


Figure 2-7. LLC PDU Control Field Formats

2.6.3 Commands and Responses

The following defines the commands and associated responses.

The C/R bit, located in the low-order bit of the SSAP, is used to distinguish between commands and responses. The following discussion of commands and responses assumes that the C/R bit has been properly decoded. The following are lists of the commands and responses (U-format only):

Unnumbered Format Commands:	Unnumbered Format Responses:
UI - Unnumbered Information	UA - Unnumbered Acknowledgement
XID - Exchange Identification	XID - Exchange Identification
TEST - Test	TEST - Test

Type 1 Operation Commands: The Type 1 commands are all U-format PDUs. The U-format PDU command encodings for Type 1 operation are listed in Figure 2-8.

FIRST CONTROL FIELD BIT DELIVERED TO/RECEIVED FROM THE MAC SUBLAYER

1	2	3	4	5	6	7	8	
1	1	0	0	P	0	0	0	UI Command
1	1	1	1	P	1	0	1	XID Command
1	1	0	0	P	1	1	1	Test Command

Figure 2-8. Type 1 Operation Command Control Field Bit Assignments

Unnumbered Information (UI) Command: The UI command PDU must be used to send information to one or more LLCs. Use of the UI command PDU is not dependent on the existence of a data link connection between the destination and source LLCs, and its use will not affect the V(S) or V(R) variables associated with any data link connections. There is no LLC response PDU to the UI command PDU.

Reception of the UI command PDU is not acknowledged or sequence number verified by the data link connection procedures; therefore, the UI PDU may be lost if a data link connection exception (such as a transmission error or a receiver-busy condition) occurs during the sending of the command PDU. A UI command PDU must have either an individual, group, global, or null address as the destination DSAP address and the originator's individual address as the SSAP address.

Exchange Identification (XID) Command: The XID command PDU must be used to convey LLC class on a per station basis, and receive window size on a per data link connection basis to the destination LLC, and to cause the destination LLC to respond with the XID response PDU (discussed below) at the earliest opportunity. The XID command PDU must have no affect on any mode or sequence numbers maintained by the remote LLC. An XID command PDU must have either an individual, group, global, or null address as the destination DSAP address and the originator's individual address as the SSAP address.

The information field of an XID basic format command PDU must consist of an 8-bit XID format identifier field plus an 8-bit parameter field that is encoded to identify the LLC class plus the receive window size, as shown in Figure 2-9. The receive window size (k) is the maximum number that the send state variable V(S) can exceed the N(R) of the last received PDU.

NOTE

Other uses of the XID PDU are for further study. In particular, the use of an unsolicited XID response PDU to announce the presence of a new LLC will be examined.

Test (TEST) Command: The TEST command PDU must be used to cause the destination LLC to respond with the TEST response PDU (discussed below) at the earliest opportunity, thus performing a basic test of the LLC to LLC transmission path. An information field is optional with the TEST command PDU. If present, however, the received information field must be returned, if possible, by the addressed LLC in the TEST response PDU. The TEST command PDU must have no affect on any mode or sequence numbers maintained by the remote LLC and may be

used with an individual, group, global or null DSAP address, and with an individual, group or global DA address.

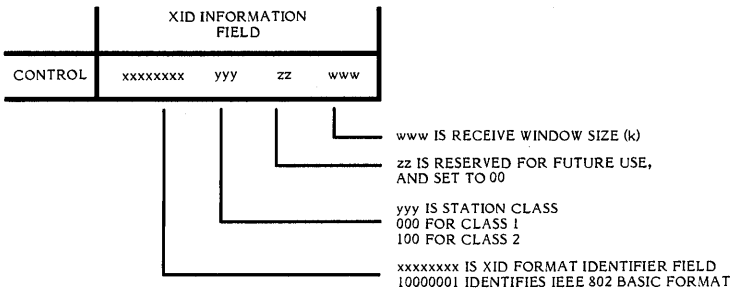


Figure 2-9. XID Information Field Basic Format

Type 1 Operation Responses: The Type 1 responses are all U-format PDUs. The U-format PDU response encodings defined for Type 1 operation are listed in Figure 2-10:

FIRST CONTROL FIELD BIT DELIVERED TO/RECEIVED FROM THE MAC SUBLAYER

1	2	3	4	5	6	7	8	
1	1	1	1	F	1	0	1	XID Response
1	1	0	0	F	1	1	1	Test Response

Figure 2-10. Type 1 Operation Response Control Field Bit Assignments

Exchange Identification (XID) Response: The XID response PDU must be used to reply to an XID command PDU at the earliest opportunity. The XID response PDU must identify the responding LLC and must include an information field like that defined for the XID command PDU (previously discussed), regardless of what information is present in the information field of the received XID command PDU. The XID response PDU must use an individual or null DSAP address, and must use an

individual or null SSAP address. The XID response PDU must have its F bit set to the state of the P bit in the XID command PDU.

Test (TEST) Response: The TEST response PDU must be used to reply to the TEST command PDU. The TEST response PDU must have its F bit set to the value of the P bit in the TEST command PDU. An information field, if present in the TEST command PDU, must be returned in the corresponding TEST response PDU. If the LLC cannot accept an information field (e.g., buffering limitation), a TEST response PDU without an information field may be returned.

2.7 LLC DESCRIPTION OF TYPE 1 PROCEDURES

In Type 1 operation, no modes of operation are defined. A LLC using Type 1 procedures must support the entire procedure set whenever it is operational on the local area network.

2.7.1 Procedure For Addressing

The address fields must be used to indicate the source (SSAP) and destination (DSAP) of the LLC PDU. The first bit in the source address field (SSAP) must be used to identify whether a command or a response is contained in the PDU.

Individual, group, global, and null addressing must be supported for destination DSAP addresses. The source address field (SSAP) must contain either an individual or null source address (see address usage in section 2.4.2).

2.7.2 Procedure For the Use of the P/F Bit

A UI command PDU must only be sent with the P bit set to "0". If a UI command PDU is received with the P bit set to "1", the LLC sublayer must optionally discard it or pass it to the higher layer with a flag identifying that the P bit was set to "1". Since a UI PDU must not be sent as a response PDU, procedures regarding the use of the F bit do not apply.

An XID command PDU must have the P bit set to either "0" or "1". Upon receipt of an XID command PDU, the receiving LLC must return an XID response PDU which has the F bit set equal to the value of the P bit contained in the incoming command PDU.

A TEST command PDU must have the P bit set to either "0" or "1". Upon receipt of a TEST command PDU, the receiving LLC must return a TEST response PDU which has the F bit set equal to the value of the P bit contained in the incoming command PDU.

2.7.3 Procedures For Logical Data Link Setup and Disconnection

Type 1 operation does not require any prior data link connection establishment (set-up), and, hence, no data link disconnection. Once the service access point has been enabled within the LLC, presumably by layer management's request, information may be sent to or received from a remote LLC service access point which is also participating in Type 1 operation.

2.7.4 Procedures For Information Transfer

Sending UI PDUs: Information transfer must be accomplished by sending the UI command PDU with the P bit set to "0". Sending UI PDUs with the P bit set to "1" or as response PDUs is prohibited. It must be possible to send the UI command PDU at any time.

Receiving UI PDUs: Reception of the UI command PDU must not be acknowledged or sequence number verified by the logical data link procedures; therefore, it will be possible for the UI PDU to be lost if a logical data link exception occurs during the sending of the command PDU. It must be possible to receive a UI command PDU at any time. However, local conditions at the receiver may result in the discarding of valid UI command PDUs by the receiving LLC. UI command PDUs which are received with the P bit set to "1" must optionally be discarded or passed to the higher layer with a flag identifying that the P bit was set to "1".

UI PDUs which are response PDUs are invalid transmissions and must be discarded by the receiving LLC.

2.7.5 Uses of the XID Command PDU and Response PDU

While response to an XID command PDU is mandatory, the origination of an XID command PDU is optional. It must be possible for the XID capabilities to be used as a part of some network control functions. As such, an XID command PDU may be sent on direction from a higher layer function, an administration function having access to the Data Link Layer, or an automatic start-up function. However, it must be possible for a more capable implementation of LLC to incorporate the use of the XID function directly to make more efficient use of the protocol.

Some possible uses of the XID capabilities include:

- The XID command PDU is a way to solicit a response from a Class I LLC or a Class II LLC. As such, it represents a basic "Are You There?" test capability.

- The XID command PDU with a group DA or group DSAP address can be used to determine the group membership. In particular, the XID command PDU with a global DA address can identify all active stations.

- A duplicate address check can be made.

- For Class II LLCs in ABM, an XID exchange can be used to identify the receive window size at each LLC for that data link connection.

- An XID exchange can identify each LLC's class.

- An LLC can announce its presence with a global DA address in an XID PDU.

2.7.6 Uses of the TEST Command PDU and Response PDU

The TEST function provides a facility to conduct loopback tests of the LLC to LLC transmission path. The initiation of the TEST function may be caused by an

administration or management entity within the Data Link Layer. Successful completion of the test consists of sending a TEST command PDU with a particular information field provided by this administration or management entity to the designated destination LLC address and receiving in return exactly the same information field in a TEST response PDU.

Implementation of the TEST command PDU is optional, but every LLC must be able to respond to a received TEST command PDU with a TEST response PDU. The length of the information field is variable from 0 to the largest size specified that each LLC on this local area network must support for normal data transfer.

It must also be possible to send even larger information fields with the following interpretations. If the receiving LLC can successfully receive and return the larger information field, it will do so. If it cannot receive the entire information field but the MAC can detect a satisfactory FCS, the LLC must discard the portion of the information field received, and may return a TEST response PDU with no information field. If the MAC cannot properly compute the FCS for the overlength information fields, the LLC must discard the portion of the information field received, and give no response. Any TEST command PDU received in error must be discarded and no response PDU sent.

In the event of failure, it is the responsibility of the administration or management entity which initiated the TEST function to determine any future actions.

2.7.7 Station Component Overview

The Station Component is responsible for handling all events that are directed to the LLC as a whole (i.e., events affecting all SAPs and connections serviced by that LLC). The Station Component must begin in the DOWN state, optionally check for a duplicate station address, and potentially enter the UP state, see Figure 2-11 and Table 2-1. The UP state of the LLC Station Component provides the enabling conditions for the operation of the Service Access Point (SAP) Components.

The Station Component must be capable of receiving and responding to the XID and TEST command PDUs. It must optionally be capable of initiating the XID command PDU, if duplicate address checking is performed by the LLC entity in a particular implementation, see Table 2-1. These PDUs must use the null DSAP address to denote that the Station Component is being referenced.

The performance of the duplicate address check requires the Station Component be prepared to receive its own XID PDUs. The definition of the MAC operation provides for the ability to simultaneously transmit and receive. Since the DA–SA in the XID PDUs used for duplicate address check, the MAC will recognize its own address and pass the PDU to the Station Component. The Station Component will respond to an XID command PDU with an XID response PDU, regardless of whether it originated from itself or a remote LLC. The Station Component provides the duplicate address check by maintaining a count of received XID response PDUs. If more than one XID response PDU is received, then at least one other identical MAC DA exists on the LAN. See Figure 2-11 and Table 2-1 for details.

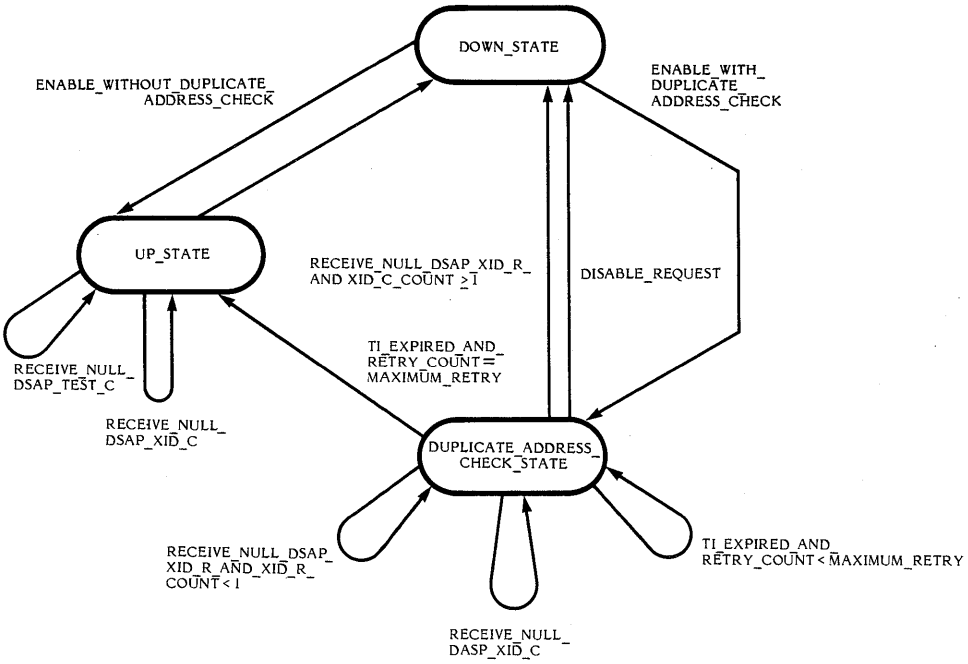


Figure 2-11. Station Component State Diagram

Table 2-1. Station Component Options

Description	States Omitted	Other Requirements
No Duplicate Address Check	DUPLICATE_ADDRESS_CHECK_STATE	Omit: ENABLE_WITH_DUPLICATE_ADDRESS_CHECK T1 EXPIRED_AND_RETRY COUNT, MAXIMUM_RETRY T1 EXPIRED_AND_RETRY COUNT=MAXIMUM_RETRY RECV_NULL_DSAP_XID_R AND_XID_R_COUNT . 1 RECV_NULL_DSAP_XID_R_ AND_XID_R_COUNT , 1
Optional Use of Duplicate Address Check	none	Omit: none
Always perform Duplicate Address Check	none	Omit: ENABLE_WITHOUT_DUPLICATE_ADDRESS CHECK

The following are descriptions of the Station Component States:

DOWN_STATE - The Station Component is powered off, not initialized, and/or disabled from operating in the local area network.

DUPLICATE_ADDRESS_CHECK_STATE - The Station Component is in the process of checking for duplicate MAC addresses on the LAN. The main purpose of this state is to allow the LLC Station Component to verify that this station's MAC address is unique on the LAN. The Station Component must send XID command PDUs with identical MAC DA and SA addresses, and must wait for a possible XID Response PDU indicating the existence of other stations with identical MAC link addresses.

UP_STATE - The Station Component is enabled, powered on, initialized and operating in the local area network. The LLC must allow SAPs to exchange LLC PDUs on the medium.

The following are Station Component Event descriptions:

ENABLE_WITH_DUPLICATE_ADDRESS_CHECK - Station Component user has initialized/enabled the station equipment, and has requested that the LLC check for MAC service access point address duplications before participating in data link communications.

ENABLE_WITHOUT_DUPLICATE_ADDRESS_CHECK - Station Component user has initialized/enabled the equipment, but duplicate MAC service access point address checking by the LLC is not supported/desired.

T1_EXPIRED_AND RETRY_COUNT,MAXIMUM RETRY - Send timer has expired and retry count is less than maximum retry limit.

T1_EXPIRED_AND RETRY_COUNT=MAXIMUM RETRY - Send timer has expired and retry count is equal to the maximum retry limit.

RECEIVE_NULL_DSAP_XID_C - An XID Command PDU with the null DSAP address has been received.

RECEIVE_NULL_DSAP XID_R_AND_XID_R_COUNT,1 - A single XID response PDU with the null DSAP address has been received.

RECEIVE_NULL_DSAP_XID_R_AND_XID_R_COUNT.1 - One or more XID response PDUs with the null DSAP address have been received.

RECEIVE_NULL_DSAP TEST_C - A TEST command PDU with the null DSAP address has been received.

DISABLE_REQUEST - Station User has requested that the equipment be disabled from operating on the medium.

The following are Station Component Action descriptions:

START_T1 - Start the send timer. This allows the LLC to determine that it has not received an acknowledgement from the remote station within a specified response time.

RETRY_COUNT:=0 - Initialize the retry counter..

RETRY_COUNT:=RETRY_COUNT+1 - Increment the retry counter.

XID_R_COUNT:=0 - Initialize the XID response PDU counter.

XID R_COUNT:=XID R_COUNT+1 - Increment the XID response PDU counter.

SEND NULL_DSAP_XID_C - The LLC must send an XID command PDU with null SSAP and null DSAP addresses and with identical MAC DA and SA addresses.

SEND XID_R - The LLC must send an XID response PDU, using the SSAP address of the XID command PDU as the DSAP address of the response PDU, and using a null SSAP address.

SEND_TEST_R - The LLC must send a TEST response PDU, using the SSAP address of the TEST command PDU as the DSAP address of the response PDU, and using a null SSAP address.

REPORT STATUS - The LLC must be able to report data link status conditions with the following valid reasons:

STATION_UP - LLC entity is now operational.

STATION_DOWN - The LLC entity is now non-operational.

DUPLICATE_ADDRESS_FOUND - LLC entity has detected another LLC entity on the LAN with a MAC service access point address identical to its own.

2.7.8 Service Access Point (SAP) Component Overview

The Service Access Point (SAP) Component handles all LLC Type 1 PDU traffic for a particular DSAP address in the local Station Component. The local service access point user is able to activate and deactivate the operation of each individual SAP Component in the Station Component (see Figure 2-12 and Table 2-2). Once active, the SAP Component must process Type 1 LLC PDUs addressed to the DSAP and send Type 1 LLC PDUs either by service access point user request or as a result of some LLC protocol action.

The following are Service Access Point (SAP) Component State descriptions:

INACTIVE_STATE - LLC SAP Component is not active, functioning, or operational. No PDUs are accepted and/or sent.

ACTIVE_STATE - LLC SAP Component is active, functioning, and operational. PDUs are received and sent.

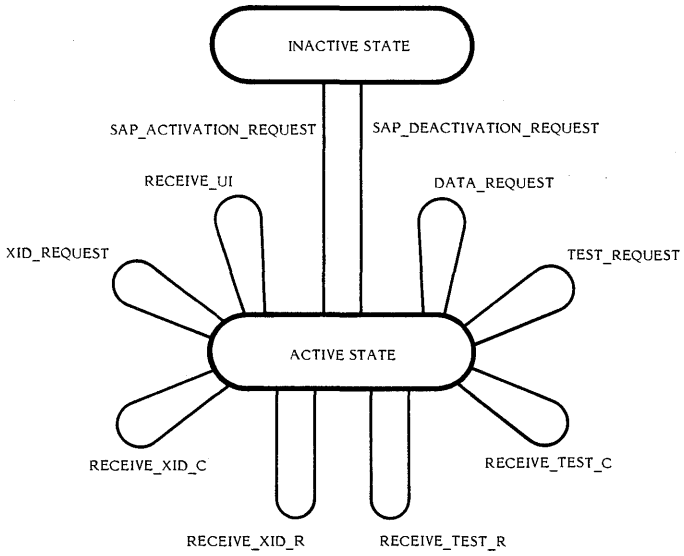


Figure 2-12. Link Service Access Point State Diagram

Table 2-2. Service Access Point Component State Transition

Current State	Event	Action(s)	Next State
INACTIVE-STATE	SAP_ACTIVATION_REQUEST	REPORT_STATUS(SAP_ACTIVE)	ACTIVE_STATE
ACTIVE_STATE	RECEIVE_UI	DATA_INDICATE	ACTIVE_STATE
	DATA_REQUEST	SEND_UI	ACTIVE_STATE
	XID_REQUEST	SEND_XID_C	ACTIVE_STATE
	RECEIVE_XID_C	SEND_XID_R	ACTIVE_STATE
	RECEIVE_XID_R	XID_INDICATE	ACTIVE_STATE
	TEST_REQUEST	SEND_TEST_C	ACTIVE_STATE
	RECEIVE_TEST_C	SEND_TEST_R	ACTIVE_STATE
	RECEIVE_TEST_R	TEST_INDICATE	ACTIVE_STATE
	SAP_DEACTIVATION_REQUEST	REPORT_STATUS(SAP_INACTIVE)	INACTIVE_ STATE

The following are Service Access Point (SAP) Component Event descriptions:

SAP_ACTIVATION REQUEST - The SAP user has requested that the particular LLC SAP Component be activated and begin logical data link operation of the Type 1 services.

SAP_DEACTIVATION_REQUEST - The SAP user has requested that the particular LLC SAP Component be deactivated and no longer allowed to operate on the logical data link.

XID_REQUEST - The SAP user has requested that the LLC SAP Component send an XID command PDU to one or more remote SAPs.

TEST_REQUEST - The SAP user has requested that the LLC SAP Component send a TEST command PDU to one or more remote SAPs.

RECEIVE_UI - The local SAP Component has received a UI PDU from a remote SAP.

DATA_REQUEST - The SAP user has requested that a Data Unit be passed to a remote LLC SAP via an MUI PDU.

RECEIVE_XID_C - The local SAP Component has received an XID command PDU from a remote SAP.

RECEIVE_XID_R - The local SAP Component has received an XID response PDU from a remote SAP.

RECEIVE_TEST_C - The local SAP Component has received a TEST command PDU from the remote SAP.

RECEIVE_TEST_R - The local SAP Component has received a TEST response PDU from the remote SAP.

The following are Service Access Point (SAP) Component Action descriptions:

DATA_INDICATE - LLC SAP Component has received a UI PDU from a remote SAP. The service data unit is given to the SAP user.

SEND_UI - A UI PDU is sent to one or more remote SAPs in response to a user request to send a service data unit.

SEND_XID_C - LLC SAP Component must send an XID command PDU to remote SAPs in response to a SAP user request to identify other SAPs.

SEND_XID_R - LLC SAP Component must send an XID response PDU to remote SAPs in response to a received XID command PDU.

SEND_TEST_C - LLC SAP Component must send a TEST command PDU in response to SAP user request to test remote SAP.

SEND_TEST R - LLC SAP Component must send a TEST response PDU in response to a remote LLC TEST command PDU.

REPORT STATUS - The LLC SAP Component must be able to report data link status conditions for the particular SAP Component with the following valid reasons:

SAP_ACTIVE - The SAP_ACTIVATION_REQUEST has been successfully processed and the component is reporting that it is now operational.

SAP_INACTIVE - The SAP_DEACTIVATION_REQUEST has been successfully processed and the component is now deactivated.

XID_INDICATE - LLC SAP Component has received an XID response PDU from a remote SAP. An indication of this event is passed to the SAP user, and may also return the XID information field.

TEST_INDICATE - LLC SAP Component has received a TEST response PDU from a remote SAP. An indication of this event is passed to the SAP user, and may also return the TEST information field.

2.8 THE MAC SUBLAYER (INTRODUCTION)

This and subsequent "subsections" of this chapter specify the services provided by the Medium Access Control (MAC) sublayer to the Logical Link Control (LLC) sublayer (see Figure 2-1). The services are described in an abstract way and do not imply any particular implementation, or any exposed interface. There is not necessarily a one-to-one correspondence between the primitives and the formal procedures and interfaces described in sections 2.12 and 2.13.

2.8.1 Overview of the Service

The services provided by the MAC sublayer allow the local LLC sublayer entity to exchange LLC data_units with peer LLC sublayer entities. Optional support may be provided for resetting the MAC sublayer entity to a known state.

2.8.2 Basic Services and Options

The primitives involved are:

MA_DATA.request
MA_DATA.confirm
MA_DATA.indicate

The MA_DATA.request, MA_DATA.confirm, and MA_DATA.indicate service primitives described below are considered mandatory.

2.9 DETAILED SERVICE SPECIFICATION FOR THE MAC SUBLAYER

The following defines the primitives and services rendered through them:

MA_DATA.request

Function: This primitive defines the transfer of data from a local LLC sublayer entity to a single peer LLC entity or multiple peer LLC entities where group addressing is used.

Semantics of the Service Primitive: The semantics of the primitive are as follows:

```
MA_DATA.request (
                destination_address,
                m_sdu,
                service_class
                )
```

The destination address parameter may specify either an individual or a group MAC entity address. It must contain sufficient information to create the DA field that is appended to the frame by the local MAC sublayer entity as well as any lower level address information. The m_sdu parameter specifies the MAC service data unit to be transmitted by the MAC sublayer entity. There is sufficient information associated with m_sdu for the MAC sublayer entity to determine the length of the data unit. The service_class parameter indicates a quality of service requested by LLC or higher layer (see Additional Comments below).

When Generated: This primitive is generated by the LLC sublayer entity whenever data must be transferred to a peer LLC entity or entities. (This can be in response to a request from higher layers of protocol or from data generated internally to the LLC sublayer, such as required by Type 2 service.)

Effect of Receipt: The receipt of this primitive causes the MAC entity to append all MAC specific fields, including DA, SA, and any fields that are unique to the particular media access method, and passes the properly formed frame to the lower layers of protocol for transfer to the peer MAC sublayer entity or entities.

Additional Comments: The CSMA/CD MAC protocol provides a single quality of service regardless of the service_class requested.

MA_DATA.confirm

Function: This primitive has local significance and must provide an appropriate response to the LLC sublayer MA_DATA.request primitive signifying the success or failure of the request.

Semantics of the Service Primitive: The semantics of this primitive are as follows:

MA_DATA.confirm (transmission_status)

The transmission_status parameter is used to pass status information back to the local requesting LLC sublayer entity. It is used to indicate the success or failure (e.g., excessive collisions) of the previous associated MA_DATA.request.

When Generated: This primitive is generated in response to an MA_DATA.request from the local LLC sublayer entity.

Effect of Receipt: The effect of receipt of this primitive by the LLC sublayer is unspecified.

Additional Comments: It is assumed that sufficient information is available to the LLC sublayer to associate the response with the appropriate request (e.g., the association may be implied by the order of the responses, since the MAC sublayer requires that the requests be serviced in a first-in-first-out manner).

MA_DATA.indicate

Function: This primitive defines the transfer of data from the MAC sublayer entity to the LLC sublayer entity or entities in the case of group addresses.

Semantics of the Service Primitive: The semantics of the primitive are as follows:

```
MA_DATA.indicate (
                    destination_address,
                    source_address,
                    m_sdu,
                    reception_status
                    )
```

The destination_address parameter may be either an individual or a group address as specified by the DA field of the incoming frame. The m_sdu parameter specifies

the MAC service data unit as received by the local MAC entity. The reception_status parameter is used to pass the following status information to the peer LLC sublayer entity: reception_complete or reception_too_long.

When Generated: The MA_DATA.indicate is passed from the MAC sublayer entity to the LLC sublayer entity or entities to indicate the arrival of a frame at the local MAC sublayer entity. Such frames are reported only if they are validly formed, received without error, and their destination address designates the local MAC entity.

Effect of Receipt: The effect of receipt of this primitive by the LLC sublayer is unspecified.

Additional Comments: If the local MAC sublayer entity is designated by the destination address parameter of an MA_DATA.request, the indicate primitive will also be invoked by the MAC entity to the local LLC entity. This full duplex characteristic of the MAC sublayer may be due to unique functionality within the MAC sublayer or full duplex characteristics of the lower layers (e.g., all frames transmitted to the broadcast address invoke MA_DATA.indicates to all stations in the network, including the station that generated the request).

2.10 MEDIA ACCESS CONTROL FRAME STRUCTURE

The following defines in detail the frame structure for data communication systems using local area network media access control (MAC) procedures. It defines the relative positions of the various components of the MAC frame. It defines the method for representing station addresses. It defines a partition of the address space into individual (single station) and group (multicast or multi-station) addresses, and into user administered and globally administered addresses.

2.10.1 MAC Frame Format

Figure 2-13 shows the eight fields of a frame; the preamble, start frame delimiter, the addresses of the frame's source and destination, a length field to indicate the length of the following field, containing the LLC data to be transmitted, a field that

contains padding, if required, and the frame check sequence field containing a cyclic redundancy check value to detect transmission errors. Of these eight fields, all are of fixed size except the LLC data and PAD fields, which may contain any integral number of octets between the minimum and maximum values determined by the specific implementation of the CSMA/CD Media Access mechanism. See section 2.12 for a particular implementation.

Referring to Figure 2-13, the octets of a frame are transmitted from top to bottom, and the bits of each octet are transmitted from left to right.

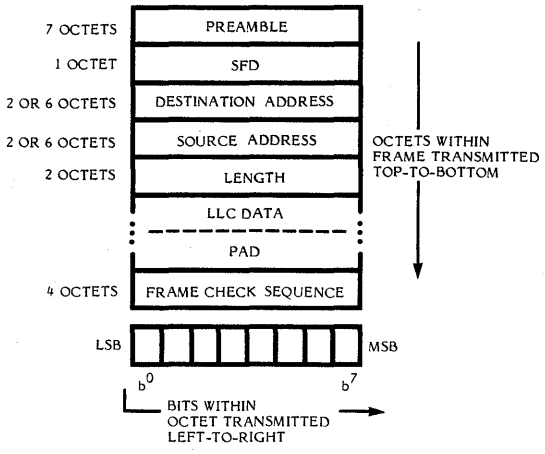


Figure 2-13. MAC Frame Format

2.10.2 Preamble Field

The preamble field is a seven octet field which is used to allow the PLS circuitry to reach its steady-state (see section 2.12.4 for preamble pattern and further details).

2.10.3 Start Frame Delimiter Field

The Start Frame Delimiter (SFD) field is the sequence 10101011. It immediately follows the preamble pattern and indicates the start of a valid frame.

2.10.4 Address Fields

Each MAC frame must contain two address fields; the Destination Address field and the Source Address field, in that order. The Destination Address field must specify the destination addressee(s) for which the frame is intended. The Source Address field must identify the station from which the frame was initiated. The representation of each address field must be as follows (see Figure 2-14):

- Each address field must contain either 16 or 48 bits. However, at any given time, the Source and Destination Address size must be the same for all stations on a particular local area network.

- The support of 16 or 48 bit address length for source and destination address is left to the manufacturer as an implementation decision. There is no requirement that manufacturers support both sizes.

- The first bit (LSB) must be used in the Destination Address field as an address type designation bit to identify the destination address either as an individual or as a group address and must not be used in the source Address field, but must be reserved and set to "0". If this bit is "0", it must indicate that the address field contains an individual address. If this bit is "1", it must indicate that the address field contains a group address that identifies none, one or more, or all of the stations connected to the local area network.

- For 48 bit addresses, the second bit must be used to distinguish between locally or globally administered addresses. For globally administered (or UPC) addresses, the bit is set to "0". If an address is to be assigned locally, this bit must be set to "1". Note that for the broadcast address, this bit is also a "1".

- Each octet of each address field must be transmitted least significant bit first.

A Media Access Control sublayer address is of one of two types:

Individual address: The address associated with a particular station on the network.

Group address: A multi-destination (multicast) address, associated with one or more stations on a given network. There are two kinds of multicast address:

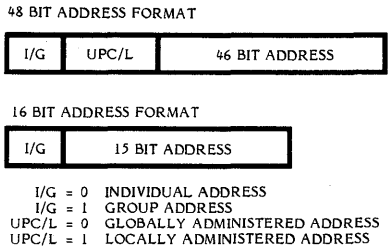


Figure 2-14. Address Field Format

Multicast-group address: An address associated by higher-level convention with a group of logically related stations.

Broadcast address: A distinguished, predefined multicast address which always denotes the set of all stations on a given local area network.

All 1's in the destination address field (for 16 or 48 bit address size LAN's) is predefined to be the Broadcast address. This group is predefined for each communication medium to consist of all stations actively connected to that medium; it must be used to broadcast to all the active stations on that medium. All stations must be able to recognize the Broadcast address. It is not necessary that a station be capable of generating the Broadcast address.

The address space must also be partitioned into locally administered and globally administered addresses.

2.10.5 Destination Address Field

The destination address field specifies the station(s) for which the frame is intended. It may be an individual or multicast (including broadcast) address.

2.10.6 Source Address Field

The source address field specifies the station sending the frame. The source address field is not interpreted by the CSMA/CD Media Access sublayer.

2.10.7 Length Field

The length field is a 2 octet field whose value indicates the number of LLC data octets in the data field. If the value is less than the minimum required for proper operation of the protocol, a pad, which is a sequence of octets, will be added at the end of the data field but prior to the FCS field, specified below. The procedure which determines the size of the pad field is specified in section 2.12.7. The length field is transmitted and received high order octet first.

2.10.8 Data and PAD Fields

The data field contains a sequence of n octets. Full data transparency is provided, in the sense that any arbitrary sequence of octet values may appear in the data field up to a maximum limit specified by the particular implementation of this standard (see section 2.14). If the frameSize is less than minFrameSize (see section 2.12.2), then the data field is extended by appending extra bits, or a pad, in units of octets, after the end of the LLC data field but prior to calculating and appending the FCS.

2.10.9 Frame Check Sequence Field

A Cyclic Redundancy Check (CRC) is used by both the transmit and receive algorithms to generate a CRC value for the FCS field.

The frame check sequence (FCS) field contains a 4-octet (32-bit) cyclic redundancy check (CRC) value. This value is computed as a function of the contents of the source, destination, length, LLC data, and pad (i.e., all fields except the preamble, SFD, and FCS). The encoding is defined by the following generating polynomial, where the symbol **N represents an exponent:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} +$$

$$x^{**}8 + x^{**}7 + x^{**}5 + x^{**}4 + x^{**}2 + x + 1$$

Mathematically, the CRC value corresponding to a given frame is defined by the following procedure:

- The first 32 bits of the frame are complemented.
- The n bits of the frame are then considered to be the coefficients of a polynomial M(x) of degree n-1. (The first bit of the destination address field corresponds to the x**0 term.)
- M(x) is multiplied by x**32 and divided by G(x), producing a remainder R(x) of degree less than 31.
- The coefficients of R(x) are considered to be a 32-bit sequence.
- The bit sequence is complemented and the result is the CRC.

The 32 bits of the CRC value are placed in the frame check sequence field so that the x**31 term is the leftmost bit of the first octet, and the x**0 term is the rightmost bit of the last octet. (The bits of the CRC are thus transmitted in the order x**31, x**30,...,x**1, x**0.)

2.10.10 Order of Bit Transmission

Each octet of the MAC frame, with the exception of the FCS, is transmitted low-order bit first.

2.10.11 Invalid MAC Frame

An invalid MAC frame is defined as one which meets at least one of the following conditions:

- It is identified as such by the Physical Layer.
- It is not an integral number of octets in length.
- It does not contain two address fields, a length field, an FCS field, and a MAC information field (when present) in their proper order.
- The bits of the incoming frame (exclusive of the FCS field itself) do not generate a CRC value identical to the one received.

The contents of invalid MAC frames must not be passed to LLC. The occurrence of invalid MAC frames may be communicated to network management.

2.11 FUNCTIONAL MODEL OF THE IEEE 802.3 CSMA/CD ACCESS METHOD

The Media Access sublayer defines a medium-independent facility, built on the medium-dependent physical facility provided by the Physical Layer, and under the access-layer-independent local network Logical Link Control (LLC) sublayer. It is applicable to a general class of local area broadcast media suitable for use with the medium access discipline known as carrier-sense multiple-access with collision-detection (CSMA/CD).

The LLC sublayer and the Media Access sublayer together are intended to have the same function as that described in the OSI model for the data link layer alone. In a broadcast network, the notion of a data link between two network entities does not correspond directly to a distinct physical connection. Nevertheless, the partitioning of functions presented in this standard requires two main functions generally associated with a data link control procedure to be performed in the Media Access sublayer. They are:

Data encapsulation (Transmit-and Receive - Data Encapsulation)

- framing (frame boundary delimitation, frame synchronization)
- addressing (handling of source and destination addresses)
- error detection (detection of physical medium transmission errors)

Medium Access Management

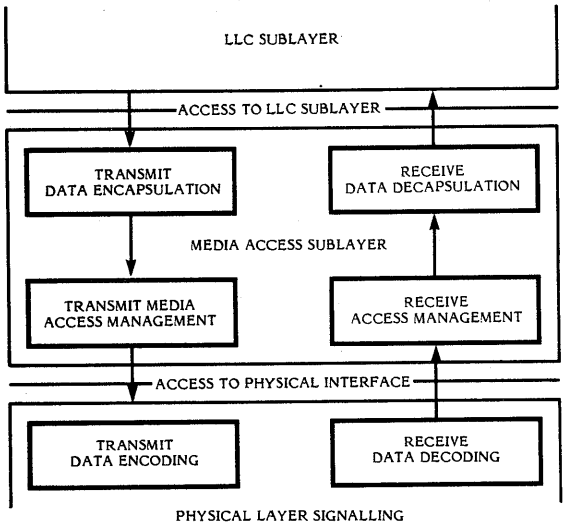
- medium allocation (collision avoidance)
- contention resolution (collision handling)

The following sections provide a functional model of the CSMA/CD Media Access method.

2.11.1 IEEE 802 CSMA/CD Operation

The following gives an overview of frame transmission and reception in terms of the functional model of the architecture. This overview is descriptive, rather than definitional; the formal specifications of the operations described here are given in sections 2.12 and 2.13. Specific implementations for CSMA/CD mechanisms that meet these specifications are given in section 2.15. Figure 2-15 illustrates the architectural model described functionally below.

The Physical Layer Signalling (PLS) component of the Physical Layer provides an interface to the Media Access sublayers for the serial transmission of bits onto the physical media. For completeness, in the operational description below, some of these functions are included as descriptive material. The concise specification of these functions is given in Section 4.2 of IEEE Draft Standards 802.3, Revision D, December 1982.



NOTE

For more details, refer to functions listed in section 2.11.3.

Figure 2-15. (IEEE 802.3 CSMA/CD) Media Access Control Functions

Transmit frame operations are independent of the receive frame operations. A transmitted frame addressed to the originating station is received and passed to the LLC sublayer at that station. This characteristic of the MAC sublayer may be implemented by functionality with the MAC sublayer or full C duplex characteristics of portions of the lower layers.

Transmission Without Contention (Normal Operation): When a LLC sublayer requests the transmission of a frame, the Transmit Data Encapsulation component of the CSMA/CD Media Access sublayer constructs the frame from the LLC-supplied data. It appends a preamble and a start of frame delimiter to the beginning of the frame. Using information passed by the LLC sublayer, the CSMA/CD Media Access sublayer also appends a PAD at the end of the MAC information field of sufficient length to ensure that the transmitted frame length satisfies a minimum frame size requirement, see section 2.13.2. It also appends destination and source addresses, a length count field and a frame check sequence to provide for error detection. The frame is then handed to the Transmit Media Access Management component in the Media Access sublayer for transmission.

Transmit Media Access Management then attempts to avoid contention with other traffic on the medium by monitoring the carrier sense signal provided by the Physical Layer Signalling (PLS) component and deferring to passing traffic. When the medium is clear, frame transmission is initiated (after a brief interframe delay to provide recovery time for other CSMA/CD Media Access sublayers and for the physical medium). The Media Access sublayer then provides a serial stream of bits to the PLS interface for transmission.

The PLS performs the task of actually generating the electrical signals on the medium which represent the bits of the frame. Simultaneously, it monitors the medium and generates the collision detect signal which, in the contention-free case under discussion, remains off for the duration of the frame.

When transmission has completed without contention, the CSMA/CD Media Access sublayer so informs the LLC sublayer using the LLC-Media Access interface and awaits the next request for frame transmission.

Reception Without Contention (Normal Operation): At each receiving station, the arrival of a frame is first detected by the PLS, which responds by synchronizing with the incoming preamble, and by turning on the carrier sense signal. As the encoded bits arrive from the medium, they are decoded and translated back into binary data. It then passes subsequent bits up to the Media Access sublayer, where first, the leading bits are discarded, up to and including the end of the preamble and start frame delimiter.

Meanwhile, the Receive Media Access Management component of the Media Access sublayer, having seen carrier sense go on, is waiting for the incoming bits to be delivered. Receive Media Access Management collects bits from the PLS as long as the carrier sense signal remains on. When the carrier sense signal goes off, the frame is truncated to an octet boundary, if necessary, and passed to Receive Data Decapsulation for processing.

Receive Data Decapsulation checks the frame's destination address field to decide whether the frame should be received by this station. If so, it passes the destination address (DA), the source address (SA), and the LLC data unit (L_SDU) to the LLC sublayer along with an appropriate status code indicating reception complete or reception_too_long. It also checks for invalid MAC frames by inspecting the frame check sequence to detect any damage to the frame enroute, and by checking for proper octet-boundary alignment of the end of the frame. Frames with a valid FCS may also be checked for proper octet-boundary alignment.

Access Interference and Recovery: If multiple stations attempt to transmit at the same time, it is possible for their transmitting Media Access sublayers to interfere with each others' transmissions, in spite of attempts to avoid this by 'deferring'. When two stations' transmissions overlap, the resulting contention is called a collision. A given station can experience a collision during the initial part of its transmission (the "collision window"), before its transmitted signal has had time to propagate to all stations on the CSMA/CD medium and the effects of that signal to propagate back. Once the collision window has passed, the station is said to have acquired the medium; subsequent collisions are avoided since all other (properly functioning) stations can be assumed to have noticed the signal (via carrier sense) and to be deferring to it. The time to acquire the medium is thus based on the

round-trip propagation time of the physical layer whose elements include the PLS, PMA, and physical media.

In the event of a collision, the PLS of a transmitting station's Physical Layer first notices the interference on the medium and turns on the collision detect signal. This is noticed in turn by the Transmit Media Access Management component of the Media Access sublayer, and collision handling begins. First, Transmit Media Access Management must enforce the collision by transmitting a bit sequence called the jam. In section 2.14, an implementation which uses this enforcement procedure is provided. This insures that the duration of the collision is sufficient to be noticed by the other transmitting station(s) involved in the collision. After the jam is sent, Transmit Media Access Management terminates the transmission and schedules a retransmission attempt for a randomly selected time in the near future. Retransmission is attempted repeatedly in the face of repeated collisions. Since repeated collisions indicate a busy medium, however, Transmit Media Access Management attempts to adjust to the medium load by backing off (voluntarily delaying its own retransmissions to reduce its load on the medium). This is accomplished by expanding the interval from which the random retransmission time is selected on each retransmission attempt. Eventually, either the transmission succeeds, or the attempt is abandoned on the assumption that the medium has failed or has become overloaded.

At the receiving end, the bits resulting from a collision are received and decoded by the PLS just as are the bits of a valid frame. Fragmentary frames received during collisions are distinguished from valid transmissions by the Media Access Sublayer's Receive Media Access Management component.

2.11.2 Relationships to LLC Sublayer and Physical Layer

The CSMA/CD Media Access sublayer provides services to the LLC sublayer required for the transmission and reception of frames. Access to these services is specified in section 2.13. The CSMA/CD Media Access sublayer makes a best effort to acquire the medium and transfer a serial stream of bits to the PLS. Although certain errors are reported to the LLC, error recovery is not provided by MAC. Error recovery may be provided by the LLC or higher sublayers.

2.11.3 CSMA/CD Access Method Functional Capabilities

The following is a summary of the functional capabilities of the (IEEE 802 CSMA/CD) Media Access sublayer. It is intended as a quick reference guide to the capabilities of the standard:

1. For Frame Transmission
   a) Accepts data from the LLC sublayer and constructs a frame.
   b) Presents a bit-serial data stream to the physical layer for transmission on the medium. This assumes that data passed from the LLC sublayer are octet multiples.

2. For Frame Reception
   a) Receives a bit-serial data stream from the physical layer.
   b) Presents to the LLC sublayer frames that are either broadcast frames or directly addressed to the local station.
   c) All frames not addressed to the receiving station are either discarded or passed to Network Management.

3. Defers transmission of a bit-serial stream whenever the physical medium is busy.

4. Appends proper FCS value to outgoing frames and verifies full octet boundary alignment.

5. Checks incoming frames for transmission errors via FCS and verifies full octet boundary alignment.

6. Delays transmission of frame bit stream for specified interframe gap period.

7. Aborts transmission when collision is detected.

8. Schedules retransmission after a collision until a specified retry limit is exceeded.

9. Enforces existing collision to ensure propagation throughout network by sending "jam" message to medium via physical layer.

10. Discards received transmissions that are less than a minimum length.

11. Appends preamble, start frame delimiter, Destination Address, Source Address, length count, and FCS to all frames, and appropriate padding for frames whose LLC data length is less than a minimum value.

12. Removes preamble, start frame delimiter, DA, SA, length, FCS, and padding (if necessary) from received frames.

2.12 IEEE 802.3 CSMA/CD MEDIA ACCESS METHOD (FORMAL SPECIFICATION)

A precise algorithmic definition is given in the following, providing a procedural model for the (IEEE 802.3 CSMA/CD) Media Access Method in the form of a program in the language Pascal. Note that whenever there is any apparent ambiguity concerning the definition of some aspect of the (IEEE 802 CSMA/CD) Media Access Method, it is the Pascal procedural specification in sections 2.12.6 through 2.12.9 which should be consulted for the definitive statement. Sections 2.12.1 through 2.12.7 provide, in prose, a description of the access mechanism, along with the formal terminology to be used in the remaining subsections.

2.12.1 Overview of the Procedural Model

The functions of the (IEEE 802.3 CSMA/CD) Media Access Method are presented below, modeled as a program written in the language Pascal. This procedural model is intended as the primary specification of the functions to be provided in any (IEEE 802 CSMA/CD) Media Access sublayer implementation. It is important to distinguish, however, between the model and a real implementation. The model is optimized for simplicity and clarity of presentation, while any realistic implementation must place heavier emphasis on such constraints as efficiency and suitability to a particular implementation technology or computer architecture. In this context, several important properties of the procedural model must be considered:

- First, it must be emphasized that the description of the Media Access sublayer in a programming language is in no way intended to imply that procedures must be implemented as a program executed by a computer. The implementation may consist of any appropriate technology including hardware, firmware, software, or any combination.
- Similarly, it must be emphasized that it is the behavior of Media Access sublayer implementations that must match the standard, not their internal structure. The internal details of the procedural model are useful only to the extent that they help specify that behavior clearly and precisely.

- The handling of incoming and outgoing frames is rather stylized in the procedural model, in the sense that frames are handled as single entities by most of the Media Access sublayer and are only serialized for presentation to the Physical Layer. In reality, many implementations will instead handle frames serially on a bit, octet or word basis. This approach is not reflected in the procedural model, since this would only complicate the description of the functions without changing them in any way.

- The model consists of algorithms designed to be executed by a number of concurrent processes; these algorithms collectively implement the (IEEE 802.3 CSMA/CD) procedure. The timing dependencies introduced by the need for concurrent activity are resolved in two ways:

  - Processes versus External events: It is assumed that the algorithms are executed "very fast" relative to external events in the sense that a process never falls behind in its work and fails to respond to an external event in a timely manner. For example, when a frame is to be received, it is assumed that the Media Access procedure Receive Frame is always called well before the frame in question has started to arrive.

  - Processes versus Processes: Among processes, no assumptions are made about relative speeds of execution. This means that each interaction between two processes must be structured to work correctly independent of their respective speeds. Note, however, that the timing of interactions

among processes is often, in part, an indirect reflection of the timing of external events, in which case appropriate timing assumptions may still be made.

It is intended that the concurrency in the model reflect the parallelism intrinsic to the task of implementing the (IEEE 802) LLC and Media Access procedures, although the actual parallel structure of the implementations is likely to vary.

Several observations need to be made about the way in which Pascal is used for the model, including:

- Some limitations of the language have been circumvented in order to simplify the specification:

  - The elements of the program (variables, procedures, etc.) are presented in logical groupings, in top-down order. Certain Pascal ordering restrictions are thus circumvented to improve readability.

  - The process and cycle constructs of the Pascal derivative Concurrent Pascal have been introduced to indicate the sites of autonomous concurrent activity. As used here, a process is simply a parameterless procedure that begins execution at "the beginning of time" rather than being invoked by a procedure call. A cycle statement represents the main body of a process and is executed repeatedly forever.

  - The lack of variable array bounds in the language is circumvented by treating frames as if they are always of a single fixed size (which is never actually specified). In fact, of course, the size of a frame depends on the size of its data field, hence the value of the "pseudo-constant" frameSize should be thought of as varying in the long-term, even though it is fixed for any given frame.

  - The use of a variant record to represent a frame (both as fields and as bits) follows the letter but not the spirit of the Pascal Report, since it allows the underlying representation to be viewed as two different data types.

- The model makes no use of any explicit interprocess synchronization primitives. Instead, all interprocess interaction is done via carefully stylized manipulation of shared variables. For example, some variables are set by only one process and inspected by another process in such a manner that the net result is independent of their execution speeds. While such techniques are not generally suitable for the construction of large concurrent programs, they simplify the model and more nearly resemble the methods appropriate to the most likely implementation technologies (e.g., microcode, hardware state-machines, etc.).

The procedural model used here is based on five cooperating concurrent processes. Three are actually defined in the Media Access sublayer. The remaining two processes are provided by the clients of the Media Access sublayer (which may include the LLC sublayer) and use the interface operations provided by the Media Access sublayer. The five processes are thus:

Frame Transmitter Process — Frame Receiver Process

Media Access Sublayer:

Bit Transmitter Process — Bit Receiver Process
Deference Process

This organization of the model is illustrated in Figure 2-16 and reflects the fact that the communication of entire frames is initiated by the client of the Media Access sublayer, while the timing of collision backoff and of individual bit transfers is based on interactions between the Media Access sublayer and the Physical-Layer-dependent bit-time.

Figure 2-16 depicts the static structure of the procedural model, showing how the various processes and procedures interact by invoking each other. Figures 2-17 and 2-18 summarize the dynamic behavior of the model during transmission and reception, focusing on the steps that must be performed, rather than the procedural structure which performs them. The usage of the shared state variables is not depicted in the figures, but is described in the comments and prose below.

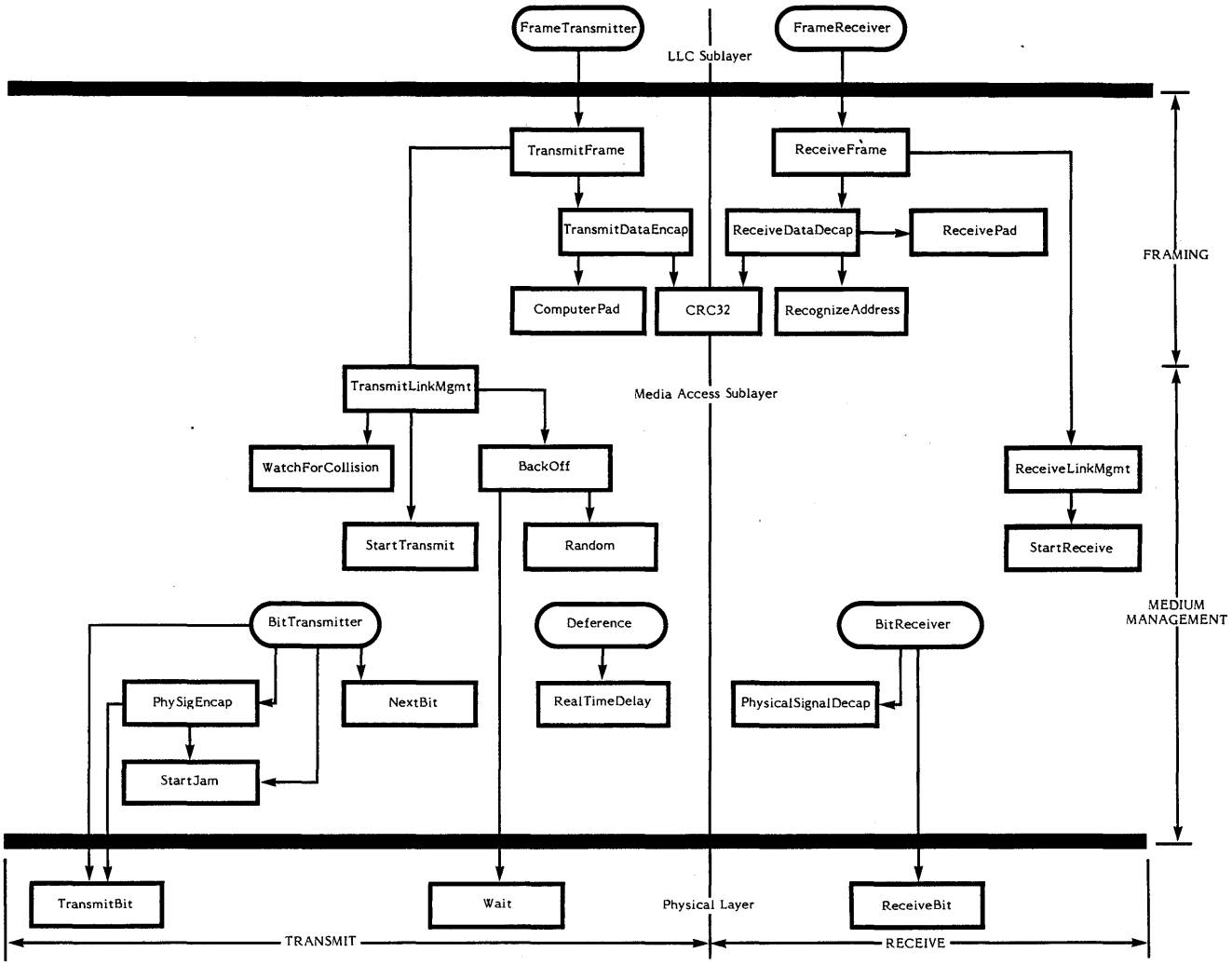


Figure 2-16. Relationship Among CSMA/CD Procedures

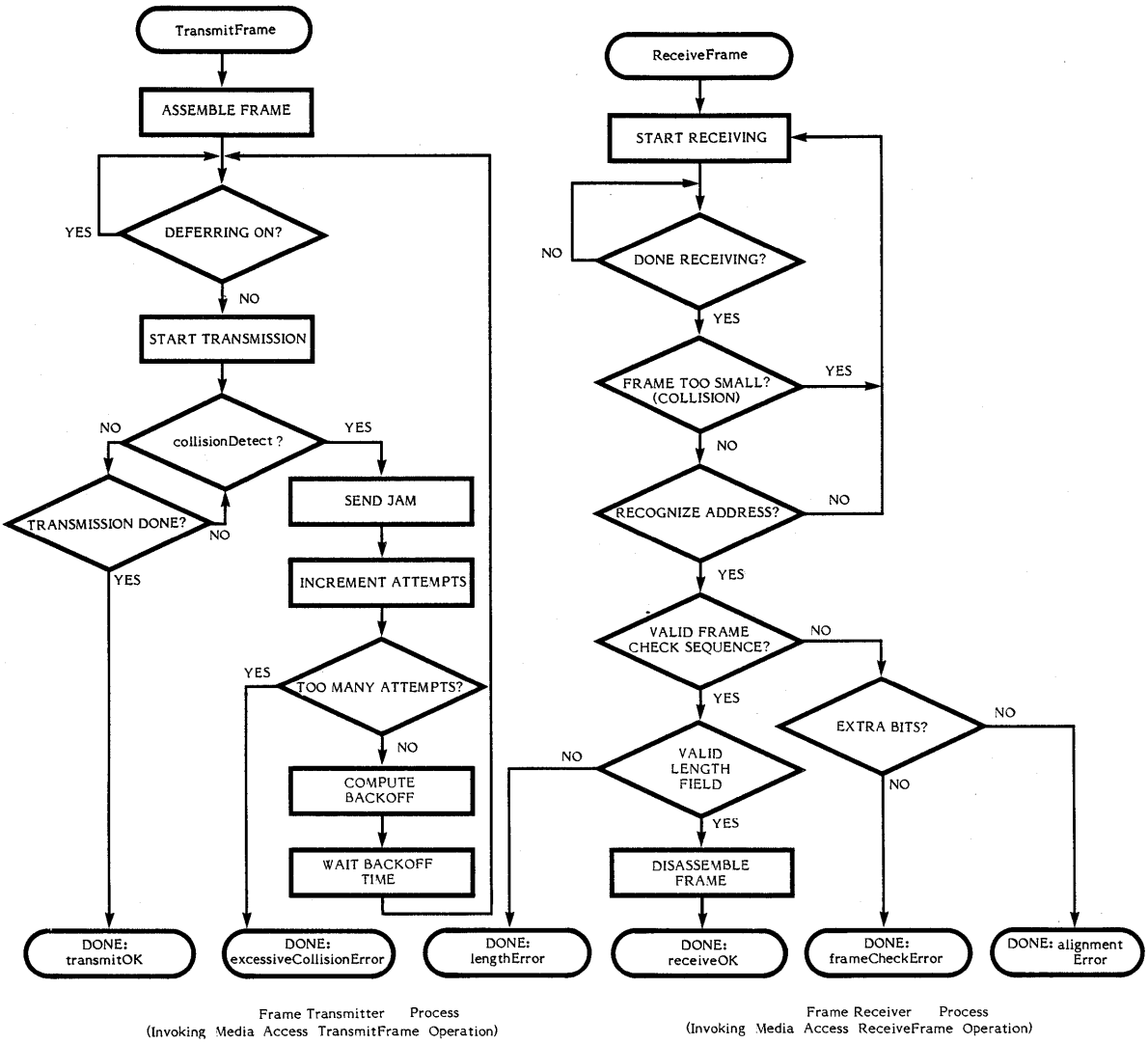


Figure 2-17. Control Flow Summary

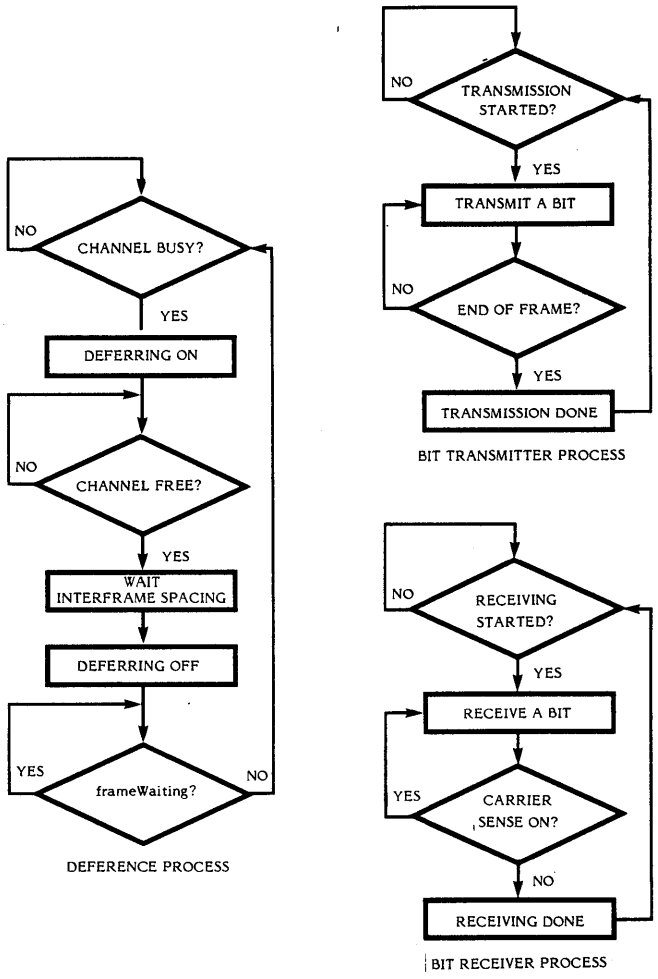


Figure 2-18. Control Flow - Media Access Sublayer

2.12.2 Frame Transmission Model

Frame transmission includes data encapsulation and Media Access management aspects:

Transmit Data Encapsulation includes the assembly of the outgoing frame (from the values provided by the LLC sublayer) and frame check sequence generation.

Transmit Media Access Management includes carrier deference, interframe spacing, collision detection and enforcement, and collision backoff and retransmission.

The following discusses Transmit Data Encapsulation:

Frame Assembly: The fields of the CSMA/CD Media Access frame are set to the values provided by the LLC sublayer as arguments to the TransmitFrame operation (see section 2.13), with the exception of the padding necessary to enforce the minimum frame size, and the frame check sequence, which is set to the CRC value generated by the Media Access sublayer.

Frame Check Sequence Generation: The CRC value defined in section 2.10.9 is generated and inserted in the frame check sequence field, following the fields supplied by the LLC Sublayer.

The following discusses Transmit Media Access Management:

Carrier Deference: Even when it has nothing to transmit, the CSMA/CD Media Access sublayer monitors the physical medium for traffic by watching the carrierSense signal provided by the PLS. Whenever the medium is busy, the CSMA/CD Media Access sublayer defers to the passing frame by delaying any pending transmission of its own. After the last bit of the passing frame (i.e., when carrierSense changes from true to false), the CSMA/CD Media Access sublayer continues to defer for a proper interframe spacing, interFrameSpacing (discussed below).

At the end of the interframe spacing of that time, if it has a frame waiting to be transmitted, transmission is initiated independent of the value of carrierSense. When transmission is completed (or immediately if there was nothing to transmit), the CSMA/CD Media Access sublayer resumes its original monitoring of carrierSense.

When a frame is submitted by the LLC sublayer for transmission, the transmission is initiated as soon as possible, but in conformance with the rules of deference stated above.

Interframe Spacing: As defined above, the rules for deferring to passing frames insure a minimum interframe spacing of interFrameSpacing seconds. This is intended to provide interframe recovery time for other CSMA/CD sublayers and for the physical medium.

Note that interFrameSpacing is the minimum value of the interframe spacing. If necessary for implementation reasons, a transmitting sublayer may use a larger value with a resulting decrease in its throughput. The larger value is determined by the parameters of the implementation, see section 2.15.

Collision Handling: Once a CSMA/CD sublayer has finished deferring and has started transmission, it is still possible for it to experience contention for the medium. Collisions can occur until acquisition of the network has been accomplished through the deference of all other stations' CSMA/CD sublayers.

The dynamics of collision handling are largely determined by a single parameter called the slot time. This single parameter describes three important aspects of collision handling:

- It is an upper bound on the acquisition time of the medium.
- It is an upper bound on the length of a frame fragment generated by a collision.
- It is the scheduling quantum for retransmission.

In order to fulfill all three functions, the slot time must be larger than the sum of the Physical Layer round-trip propagation time and the Media Access Layer maximum jam time. The slot time is determined by the parameters of the implementation, see section 2.14.

Collision Detection and Enforcement: Collisions are detected by monitoring the collisionDetect signal provided by the Physical Layer. When a collision is detected during a frame transmission, the transmission is not terminated immediately. Instead, the transmission continues until additional bits specified by JamSize have been transmitted (counting from the time collisionDetect went on). This collision enforcement, or "jam", guarantees that the duration of the collision is sufficient to ensure its detection by all transmitting stations on the network. The content of the jam is unspecified; it may be any fixed or variable pattern convenient to the Data Media Access implementation. It is recommended that it not be the 32-bit CRC value corresponding to the (partial) frame transmitted prior to the jam.

Collision Backoff and Retransmission: When a transmission attempt has terminated due to a collision, it is retried by the transmitting CSMA/CD sublayer until either it is successful, or a maximum number of attempts, attemptLimit, have been made and all have terminated due to collisions. Note that all attempts to transmit a given frame are completed before any subsequent outgoing frames are transmitted. The scheduling of the retransmissions is determined by a controlled randomization process called "truncated binary exponential backoff". At the end of enforcing a collision (jamming), the CSMA/CD sublayer delays before attempting to retransmit the frame. The delay is an integral multiple of slotTime. The number of slot times to delay before the nth retransmission attempt is chosen as a uniformly distributed random integer r in the range $0 \leq r < 2^{**}k$, where k = min (n, 10). If all attemptLimit attempts fail, this event is reported as an error. Algorithms used to generate the integer r should be designed to minimize the correlation between the numbers generated by any two stations at any given time.

Note that the values given above define the most aggressive behavior that a station may exhibit in attempting to retransmit after a collision. In the course of implementing the retransmission scheduling procedure, a station may introduce extra delays which degrade its own throughput, but in no case may a station's

retransmission scheduling result in a lower average delay between retransmission attempts than the procedure defined above.

Minimum Frame Size: The CSMA/CD Media Access mechanism requires that a minimum frame length of minFrameSize bits be transmitted. If frameSize is less than minFrameSize, then the CSMA/CD Medium Access sublayer must append extra bits, or a pad, in units of octets, after the end of the LLC data field but prior to calculating, and appending, the FCS. The number of pad units must be sufficient to ensure that the Media Access sublayer frame is at least minFrameSize bits. The pad is determined using the size of the LLC protocol data-unit. The content of the pad is unspecified.

2.12.3 Frame Reception Model

CSMA/CD Media Access sublayer frame reception includes both data decapsulation and Media Access management aspects:

Receive Data Decapsulation comprises address recognition, frame check sequence validation, and frame disassembly to pass the fields of the received frame to the LLC sublayer.

Receive Media Access Management comprises recognition of collision fragments from incoming frames and truncation of frames to octet boundaries.

The following discusses receive data decapsulation:

Address Recognition: The CSMA/CD sublayer is capable of recognizing individual and group addresses.

Individual Addresses: The CSMA/CD sublayer recognizes and accepts any frame whose destination field contains the individual address of the station.

Group Addresses: The CSMA/CD Media Access sublayer recognizes and accepts any frame whose destination field contains the broadcast address.

The CSMA/CD sublayer is capable of activating some number of group addresses as specified by higher layers. The CSMA/CD sublayer recognizes and accepts any frame whose destination field contains an active group address. An active group address may be deactivated.

Frame Check Sequence Validation: FCS validation is essentially identical to FCS generation. If the bits of the incoming frame (exclusive of the FCS field itself) do not generate a CRC value identical to the one received, an error has occurred and the frame is identified as invalid.

Frame Disassembly: Upon recognition of the start frame delimiter at the end of the preamble sequence, the CSMA/CD Media Access sublayer accepts the frame. If there are no errors, the frame is disassembled and the fields are passed to the LLC sublayer via the output parameters of the ReceiveFrame operation.

The following discusses Receive Media Access Management:

Framing: The CSMA/CD sublayer recognizes the boundaries of an incoming frame by monitoring the carrierSense signal provided by the PLS. There are two possible length errors that can occur, which indicate ill-framed data; the frame may be too long, or its length may not be an integral number of octets.

Maximum Frame Size - The receiving CSMA/CD sublayer is not required to enforce the frame size limit, but it is allowed to truncate frames longer than MAXFrameSize octets and report this event as an (implementation-dependent) error.

Integral Number of Octets in Frame - Since the format of a valid frame specifies an integral number of octets, only a collision or an error can produce a frame with a length that is not an integral multiple of 8. Complete frames (i.e., not rejected as collision fragments; see below) that do not contain an integral number of octets are truncated to the nearest octet boundary. If frame check sequence validation detects an error in such a frame, the status code alignmentError is reported.

Collision Filtering: The smallest valid frame must be at least one slotTime in length. This determines the minFrameSize. Any frame containing less than minFrameSize bits is presumed to be a fragment resulting from a collision. Since occasional collisions are a normal part of the Media Access management procedure, the discarding of such a fragment is not reported as an error to the LLC sublayer.

2.12.4 Preamble Generation

In an implementation of an IEEE 802 local area network, most of the Physical Layer components are allowed to provide valid output some time after being presented valid input. Thus, it is necessary for a preamble to be sent before the start of data, to allow the PLS circuitry to reach its steady-state. Upon request by TransmitLink-Mgmt to transmit the first bit of a new frame, PhysicalSignalEncap must first transmit the preamble, a bit sequence used for physical medium stabilization and synchronization, followed by the start frame delimiter. If, while transmitting this stream, the PLS asserts the collision detect signal, any remaining preamble bits must be sent.

The preamble pattern is:

10101010 10101010 10101010 10101010 10101010 10101010 10101010

The bits are transmitted in order, from left to right. The nature of the pattern is such that when encoded using Manchester encoding technique, it appears as a periodic waveform on the medium that enables bit synchronization. It should be noted that the preamble ends with a '0'.

2.12.5 Start Frame Sequence

The PLS recognizes the presence of activity on the medium through the carrier sense signal. This is the first indication that the frame reception process should begin. Upon reception of the sequence 10101011 immediately following a latter part of the preamble pattern, PhysicalSignalDecap shall begin passing successive bits to ReceiveLinkMgmt for passing to the LLC sublayer.

2.12.6 Global Declarations

The following provides detailed formal specifications for the (IEEE 802 CSMA/CD) Media Access Mechanism. It is a specification of generic features and parameters to be used in systems implementing this (IEEE 802) media access method. Section 2.14 provides values for these sets of parameters for recommended implementations of this media access mechanism.

Common Constants and Types: The following declarations of constants and types are used by the frame transmission and reception sections of each CSMA/CD sublayer:

```
constant
    addressSize = ... ; (16 or 48 bits in compliance with section 2.13.2)
    lengthSize = 16; (in bits)
    LLCdataSize = ...; (LLC Data, see section 2.12.1)
    padSize = ...; (in bits, = max (0, minFrameSize - LLCdataSize), see section
    2.12.1)
    dataSize = ...; (=LLCdataSize + padSize)
    crcSize = 32; (32 bit CRC = 4 octets)
    frameSize = ...; (=2* addressSize + lengthSize + dataSize + crcSize, see section
    2.12.1)
    minFrameSize = ...; (in bits, implementation dependent)
    slotTime = ...; (unit of time for collision handling, implementation dependent)
    preambleSize = ...; (in bits, physical medium dependent)
    sfdSize = 8; (8 bit start frame delimiter)
    headerSize = ...; (sum of preambleSize and sfdSize)

type
    Bit = 0..1;
    AddressValue = array (1..addressSize) of Bit;
    LengthValue = array (1..lengthSize) of Bit;
    DataValue = array (1..dataSize) of Bit;
    CRCValue = array (1..crcSize) of Bit;
```

```
PreambleValue = array (1..preambleSize) of Bit;
SfdValue = array (1..sfdSize) of Bit;
ViewPoint = (fields, bits); (Two ways to view the contents of a frame)
HeaderViewPoint = (headerFields, headerBits);
Frame = record (Format of Media Access frame)
  case view: ViewPoint of
    fields: (
      destinationField: AddressValue;
      sourceField: AddressValue;
      lengthField: LengthValue;
      dataField: DataValue;
      fcsField: CRCValue);
    bits: (contents: array (1..frameSize) of Bit)
  end; (Frame)
Header = record (Format of preamble and start frame delimiter)
  case headerView : HeaderViewPoint of
    headerFields : (
      preamble : PreambleValue;
      sfd : SfdValue);
    headerBits : (
      headerContents : array (1..headerSize) of Bit)
  end; (defines header for MAC frame)
```

Transmit State Variables: The following items are specific to frame transmission. (See also section 2.14.)

```
const
    interFrameSpacing = ... ; (minimum time between frames)
    attemptLimit = ... ; (Max number of times to attempt transmission)
    backOffLimit = ... ; (Limit on number of times to back off)
    jamSize = ... ;       (in bits: the value depends upon medium and collision detect
                           implementation)
var
    outgoingFrame: Frame; (The frame to be transmitted)
    outgoingHeader: Header;
```

```
    currentTransmitBit, lastTransmitBit: 1..frameSize;
    (Positions of current and last outgoing bits in outgoingFrame)
    lastHeaderBit: 1..headerSize;
    deferring: Boolean;          (Implies any pending transmission must wait for the
                                  medium to clear)
    frameWaiting: Boolean; (Indicates that outgoingFrame is deferring)
    attempts: 0..attemptLimit;   (Number of transmission attempts on outgoing
                                  Frame)
    newCollision: Boolean;       (Indicates that a collision has occured but has not yet
                                  been jammed)
    transmitSucceeding: Boolean; (Running indicator of whether Transmission is
                                  succeeding)
```

Receive State Variables: The following items are specific to frame reception. (See also section 2.14.)

```
var
    incomingFrame: Frame; (The frame being received)
    currentReceiveBit: 1..frameSize; (Position of current bit in incomingFrame)
    receiving: Boolean; (Indicates that a frame reception is in progress)
    excessBits: 0..7; (Count of excess trailing bits beyond octet boundary)
    receiveSucceeding: Boolean;  (Running indicator of whether reception is suc-
                                  ceeding)
    validLength: Boolean; (Indicator of whether received frame has a length error)
```

Summary of Interlayer Interfaces: The interface to the LLC sublayer, defined in section 2.14, is summarized below:

```
type

TransmitStatus = (transmitOK, excessiveCollisionError);
                 (Result of TransmitFrame operation)
ReceiveStatus = (receiveOK, lengthError, frameCheckError, alignmentError);
                 (Result of ReceiveFrame operation)
```

```
function TransmitFrame (
    destinationParam: AddressValue;
    sourceParam: AddressValue;
    lengthParam: LengthValue
    dataParam: DataValue): TransmitStatus; (Transmits one frame)

function ReceiveFrame (
    var destinationParam: AddressValue;
    var sourceParam: AddressValue;
    var lengthParam: LengthValue;
    var dataParam: DataValue): ReceiveStatus; (Receives one frame)
```

The interface to the Physical Layer, defined in section 2.14 is summarized below:

```
var
    carrierSense: Boolean; (Indicates incoming bits)
    transmitting: Boolean; (Indicates outgoing bits)
    collisionDetect: Boolean; (Indicates medium contention)
procedure TransmitBit (bitParam: Bit); (Transmits one bit)
function ReceiveBit: Bit; (Receives one bit)
procedure Wait (bitTimes: integer); (Waits for indicated number of bit-times)
```

State Variable Initialization: The procedure Initialize must be run when the Media Access sublayer begins operation, before any of the processes begin execution. Initialize sets certain crucial shared state variables to their initial values. (All other global variables are appropriately reinitialized before each use.) Initialize then waits for the medium to be idle, and starts operation of the various processes:

```
procedure Initialize;
begin
    frameWaiting := false;
    deferring := false;
    newCollision := false;
    transmitting := false; (In interface to Physical Layer; see below)
    receiving := false;
```

```
while carrierSense do nothing;
(Start execution of all processes)
end; (Initialize)
```

2.12.7 Frame Transmission

The algorithms in the following define Media Access sublayer frame transmission. The function TransmitFrame implements the frame transmission operation provided to the LLC sublayer:

```
function TransmitFrame (
    destinationParam: AddressValue;
    sourceParam: AddressValue;
    lengthParam: LengthValue;
    dataParam: DataValue): TransmitStatus;
procedure TransmitDataEncap; ... (nested procedure; see body below)
begin
    TransmitDataEncap;
    TransmitFrame := TransmitLinkMgmt
end; (TransmitFrame)
```

First, TransmitFrame calls the internal procedure TransmitDataEncap to construct the frame. Next, TransmitLinkMgmt is called to perform the actual transmission. The TransmitStatus returned indicates the success or failure of the transmission attempt.

TransmitDataEncap builds the frame and places the 32-bit CRC in the frame check sequence field:

```
procedure TransmitDataEncap;
begin

    with outgoingFrame do
    begin (assemble frame)
      view := fields;
```

```
      destinationField := destinationParam;
      sourceField := sourceParam;
      lengthField := lengthParam;
      dataField := ComputePad (lengthparam, dataParam);
      fcsField := CRC32 (outgoingFrame);
      view := bits
    end (assemble frame)
    With outgoingHeader do
    begin
      headerView: = headerFields;
      preamble: = ...; (* '1010...10', LSB to MSB*)
      sfd: = ...; (* '10101011', LSB to MSB*)
      headerView: = headerBits
    end
end; (TransmitDataEncap)
```

ComputePad appends an array of arbitrary bits to the LLCDataField to pad the frame to the minimum frame size.

```
function ComputePad (
    var  lengthParam:LengthValue
    var  dataParam:DataValue) :DataValue;
begin
    ComputePad: = (Append an array of size PadSize of arbitrary bits to the
                    LLCdataField)
end; (ComputePadParam)
```

TransmitLinkMgmt attempts to transmit the frame, deferring first to any passing traffic. If a collision occurs, transmission is terminated properly and retransmission is scheduled following a suitable backoff interval:

```
function TransmitLinkMgmt:  TransmitStatus;
begin
    attempts := 0; transmitSucceeding := false;
    while attempts , attemptLimit and not transmitSucceeding do
    begin (loop)
```

```
      if attempts . 0 then Backoff;
      frameWaiting := true;
      while deferring do nothing; (defer to passing frame, if any)
      frameWaiting := false;
      StartTransmit;
      while transmitting do WatchForCollision;
      attempts := attempts + 1
    end; (loop)
    if transmitSucceeding then TransmitLinkMgmt := transmitOK
    else TransmitLinkMgmt := excessiveCollisionError
end; (TransmitLinkMgmt)
```

Each time a frame transmission attempt is initiated, StartTransmit is called to alert the BitTransmitter process that bit transmission should begin:

```
procedure StartTransmit;
begin
    currentTransmitBit := 1;
    lastTransmitBit := frameSize;
    transmitSucceeding := true;
    transmitting := true;
    lastHeaderBit := headerSize
end; (StartTransmit)
```

Once frame transmission has been initiated, TransmitLinkMgmt monitors the medium for contention by repeatedly calling WatchForCollision:

```
procedure WatchForCollision;
begin
    if TransmitSucceeding and collisionDetect then
    begin
      newCollision := true;
      transmitSucceeding := false
    end
end; (WatchForCollision)
```

WatchForCollision, upon detecting a collision, updates newCollision to insure proper jamming by the BitTransmitter process.

After transmission of the jam has completed, if TransmitLinkMgmt determines that another attempt should be made, BackOff is called to schedule the next attempt to retransmit the frame.

```
var maxBackOff: 2..1024; (Working variable of BackOff)

procedure BackOff;
begin
  if attempts = 1 then maxBackOff := 2
  else if attempts ,= backoffLimit
  then maxBackOff := maxBackOff
  Wait (slotTime*Random(0, maxBackOff))
end; (BackOff)

function Random (low, high: integer): integer;

begin
  Random := ... (uniformly distributed random integer r such that low ≤ r ≤ high)
end; (Random)
```

BackOff performs the truncated binary exponential backoff computation and then waits for the selected multiple of the slot time.

The Deference process runs asynchronously to continuously compute the proper value for the variable deferring.

```
process Deference;
begin
  cycle (main loop)
    while not carrierSense do nothing; (watch for carrier to appear)
    deferring := true; (delay start of new transmissions)
```

```
      while carrierSense do nothing; (wait for carrier to disappear)
      RealTimeDelay (interFrameSpacing);
      deferring := false; (allow new transmissions to proceed)
      while frameWaiting do nothing (allow waiting transmission, if any)
    end (main loop)
  end; (Deference)

  procedure RealTimeDelay (usec: real);
  begin
    (Wait for the specified number of microseconds)
  end; (RealTimeDelay)
```

The BitTransmitter process runs asynchronously, transmitting bits at a rate determined by the Physical Layer's TransmitBit operation:

```
  process BitTransmitter;
  begin
    cycle (outer loop)
      If transmitting, then
      begin (inner loop)
        PhysicalSignalEncap; (Send preamble and start of frame delimiter)
        while transmitting do
        begin
          TransmitBit (outgoingFrame (currentTransmitBit)); (send next bit to
          Physical Layer)
          if newCollision, then StartJam else NextBit
        end;
      end; (inner loop)
    end; (outer loop)
  end; (BitTransmitter)

   rocedure PhysicalSignalEncap;
  begin
    while currentTransmitBit ,= lastHeaderBit do
    begin
```

```
      TransmitBit (outgoingHeader (currentTransmitBit)); (transmit header one bit
      at a time)
      currentTransmitBit := currentTransmitBit + 1;
    end
    if newCollision, then StartJam else
    currentTransmitBit := 1
end; (PhysicalSignalEncap)

procedure NextBit;
begin
    currentTransmitBit := currentTransmitBit + 1;
    transmitting := (currentTransmitBit ,= lastTransmitBit)
end; (NextBit)

procedure StartJam;
begin
    currentTransmitBit := 1;
    lastTransmitBit := jamSize;
    newCollision := false
end; (StartJam)
```

BitTransmitter, upon detecting a new collision, immediately enforces it by calling StartJam to initiate the transmission of the jam. The jam should contain a sufficient number of bits of arbitrary data so that it is assured that both communicating stations detect the collision. (StartJam uses the first set of bits of the frame up to JamSize, merely to simplify this program).

2.12.8 Frame Reception

The algorithms in this section define CSMA/CD Media Access sublayer frame reception. The procedure ReceiveFrame implements the frame reception operation provided to the LLC sublayer:

```
function ReceiveFrame (
    var destinationParam: AddressValue;
```

```
    var sourceParam: AddressValue;
    var lengthParam: LengthValue;
    var dataParam: DataValue): ReceiveStatus;
  function ReceiveDataDecap: ReceiveStatus; ... (nested function; see body below)

  begin
    repeat
      ReceiveLinkMgmt;
      ReceiveFrame := ReceiveDataDecap;
    until receiveSucceeding
  end; (ReceiveFrame)
```

ReceiveFrame calls ReceiveLinkMgmt to receive the next valid frame, and then calls the internal procedure ReceiveDataDecap to return the frame's fields to the LLC sublayer if the frame's address indicates that it should do so. The returned ReceiveStatus indicates the presence or absence of detected transmission errors in the frame.

```
  function ReceiveDataDecap: ReceiveStatus;
  begin
    receiveSucceeding := RecognizeAddress (incomingFrame. destinationField);
    if receiveSucceeding, then with incomingFrame do
    begin (disassemble frame)
      view := fields;
      destinationParam := destinationField;
      sourceParam := sourceField;
      lengthParam := lengthField;
      dataParam := RemovePad (lengthField, dataField);
      if fcsField = CRC32 (incomingFrame), then
      begin
        if validLength, then ReceiveDataDecap: = receiveOK
        else ReceiveDataDecap: = lengthError
      end
```

```
      else
      begin
        if excessBits = 0, then ReceiveDataDecap := frameCheckError
        else ReceiveDataDecap := alignmentError;
      end;
      view: = bits
    end (disassemble frame)
  end; (ReceiveDataDecap)

  function RecognizeAddress (address: AddressValue): Boolean;
  begin
    RecognizeAddress := ... (Returns true for the set of physical, broadcast, and
    multicast-group addresses corresponding to this station)
  end; (RecognizeAddress)

  function RemovePad (
    var lengthParam:LengthValue
    var dataParam:DataValue):DataValue;
  begin
    RemovePad:=(strips lengthParam bits from the data field and returns the
    LLCDataField);
    validLength:=(Check to determine if value represented by lengthParam matches
    received LLCdataSize)
  end; (RemovePad)
```

ReceiveLinkMgmt attempts repeatedly to receive the bits of a frame discarding any fragments from collisions by comparing them to the minimum valid frame size:

```
  procedure ReceiveLinkMgmt;
  begin
    repeat
      StartReceive;
      while receiving do nothing; (wait for frame to finish arriving)j
      excessBits := frameSize mod 8;
      frameSize := frameSize - excessBits; (truncate to octet boundary)
```

```
      receiveSucceeding := (frameSize .= minFrameSize); (reject collision frag
      ments)
    until receiveSucceeding
end; (ReceiveLinkMgmt)

procedure StartReceive;
begin
  currentReceiveBit := 1;
  receiving := true
end; (StartReceive)
```

The BitReceiver process runs asynchronously, receiving bits from the medium at th rate determined by the Physical Layer's ReceiveBit operation:

```
process BitReceiver;
  var b: Bit;
begin
  cycle (outer loop)
    while receiving do
    begin (inner loop)
    If currentReceiveBit = 1, then
      PhysicalSignalDecap; (Strip off the preamble and start frame delimiter)
      b := ReceiveBit; (Get next bit from physical Media Access)
      if carrierSense, then
      begin (append bit to frame)
        incomingFrame (currentReceiveBit) := b;
        currentReceiveBit := currentReceiveBit + 1
      end; (append bit to frame)
      receiving := carrierSense
    end (inner loop)
    frameSize: = currentReceiveBit - 1
  end (outer loop)
end; (Bit Receiver)
```

```
Procedure PhysicalSignalDecap;
begin
  (Receive one bit at a time from physical medium until a valid sfd is detected,
  discard bits and return)
end; (PhysicalSignalDecap)
```

2.12.9 Common Procedures

The function CRC32 is used by both the transmit and receive algorithms to generate a 32 bit CRC value:

```
function CRC32 (f: Frame): CRCValue;
begin
  CRC32 := (The 32-bit CRC)
end; (CRC32)
```

Purely to enhance readability, the following procedure is also defined:

```
procedure nothing;
begin end;
```

The idle state of a process (i.e., while waiting for some event) is cast as repeated calls on this procedure.

2.13 INTERFACES TO/FROM ADJACENT LAYERS

The purpose of the following is to provide precise definitions of the interfaces between architectural layers in compliance with the Media Access Service Specification given in section 2.2. In addition, the services required from the physical medium are defined.

The notation used here is the Pascal language, in keeping with the procedural nature of the formal and Media Access sublayers specification (see section 2.11). Each interface is described as a set of procedures and/or shared variables that collectively provide the only valid interactions between layers. The accompanying text

describes the meaning of each procedure or variable and points out any implicit interactions among them.

Note that the description of the interfaces in Pascal is a notational technique, and in no way implies that they can or should be implemented in software. This point is discussed more fully in section 2.12, that provides complete Pascal declarations for the data types used in the remainder of this section. Note also that the "synchronous" (one frame at a time) nature of the frame transmission and reception operations is a property of the architectural interface between the LLC and Media Access sublayers, and need not be reflected in the implementation interface between a station and its sublayer.

2.13.1 Services Provided by the Media Access Sublayer

Two services provided to the LLC sublayer by the Media Access sublayer are transmission and reception of LLC frames. The interface through which the LLC sublayer uses the facilities of the Media Access sublayer, therefore, consists of a pair of functions.

```
Functions:
  TransmitFrame
  ReceiveFrame
```

Each of these functions has the components of a LLC frame as its parameters (input or output), and returns a status code as its result.

The LLC sublayer transmits a frame by invoking TransmitFrame:

```
function TransmitFrame (
  destinationParam: AddressValue;
  sourceParam: AddressValue;
  lengthParam: LengthValue;
  dataParam: DataValue): TransmitStatus;
```

The TransmitFrame operation is synchronous. Its duration is the entire attempt to transmit the frame; when the operation completes, transmission has either succeeded or failed, as indicated by the resulting status code:

```
type TransmitStatus = (transmitOK, excessiveCollisionError);
```

Successful transmission is indicated by the status code transmitOK; the code excessiveCollisionError indicates that the transmission attempt was aborted due to excessive collisions, because of heavy traffic or a network failure.

The LLC sublayer accepts incoming frames by invoking ReceiveFrame:

```
function ReceiveFrame (
  var destinationParam: AddressValue;
  var sourceParam: AddressValue;
  var length Param: LengthValue;
  var dataParam: DataValue): ReceiveStatus;
```

The ReceiveFrame operation is synchronous. The operation does not complete until a frame has been received. The fields of the frame are delivered via the output parameters, along with a status code:

```
type ReceiveStatus = (receiveOK, lengthError, frameCheckError,
alignmentError);
```

Successful reception is indicated by the status code receiveOK. The code frameCheckError indicates that the frame received was damaged by a transmission error in the physical medium. The code alignmentError indicates that the frame received was damaged, and that, in addition, its length was not an integral number of octets. The lengthError indicates that the length Parameter value was inconsistent with the frameSize of the received frame.

2.13.2 Services Required From the Physical Layer

The interface through which the CSMA/CD Media Access sublayer uses the facilities of the Physical Layer consists of a function, a pair of procedures and three Boolean variables:

Function:
  ReceiveBit

Variables:
  collisionDetect
  carrierSense
  transmitting

Procedures:
  TransmitBit
  Wait

During transmission, the contents of an outgoing frame are passed from the Media Access sublayer to the Physical Layer via repeated use of the TransmitBit operation:

```
procedure TransmitBit (bitParam: Bit);
```

Each invocation of TransmitBit passes one new bit of the outgoing frame to the Physical Layer. The TransmitBit operation is synchronous. The duration of the operation is the entire transmission of the bit. The operation completes, when the Physical Layer is ready to accept the next bit and it transfers control to the MAC sublayer.

The overall event of data being transmitted is signalled to the Physical Layer via the variable transmitting:

```
var transmitting: Boolean;
```

Before sending the first bit of a frame, the Media Access sublayer sets transmitting to true, to inform the Physical Media Access that a stream of bits will be presented via the TransmitBit operation. After the last bit of the frame has been presented, the Media Access sublayer sets transmitting to false to indicate the end of the frame.

The presence of a collision in the physical medium is signalled to the Media Access sublayer by the variable collisionDetect:

```
var collisionDetect: Boolean;
```

The collisionDetect signal remains true during the duration of the collision. (Note: Since an entire collision may occur during preamble generation, the Media Access sublayer must handle this possibility by monitoring collisionDetect concurrently with its transmission of outgoing bits. See section 2.12 for details.)

The collisionDetect signal is generated only during transmission and is never true at any other time; in particular, it cannot be used during frame reception to detect collisions between overlapping transmissions from two or more other stations.

During reception, the contents of an incoming frame are retrieved from the Physical Layer by the Media Access sublayer via repeated use of the ReceiveBit operation:

```
function ReceiveBit: Bit;
```

Each invocation of ReceiveBit retrieves one new bit of the incoming frame from the Physical Layer. The ReceiveBit operation is synchronous. Its duration is the entire reception of a single bit. Upon receiving a bit, the Media Access sublayer must immediately request the next bit until all bits of the frame have been received. (See section 2.12 for details.)

The overall event of data being received is signalled to the Media Access sublayer by the variable carrierSense:

```
var carrierSense: Boolean;
```

When the Physical Layer sets carrierSense to true, the Media Access sublayer must immediately begin retrieving the incoming bits by the ReceiveBit operation. When carrierSense subsequently becomes false, the Media Access sublayer can begin processing the received bits as a completed frame. Note that the true/false transitions of carrierSense are not defined to be precisely synchronized with the beginning and end of the frame, but may precede the beginning and lag the end, respectively. If an invocation of ReceiveBit is pending when carrierSense becomes false, ReceiveBit returns an undefined value, which should be discarded by the Media Access sublayer. (See section 2.13 for details.)

The Media Access sublayer must also monitor the value of carrierSense to defer its own transmissions when the medium is busy.

The Physical Layer also provides the procedure Wait:

```
procedure Wait (bitTimes: integer);
```

This procedure waits for the specified number of bit times. This allows the Media Access sublayer to measure time intervals in units of the (physical-medium-dependent) bit time.

Another important property of the Physical Layer which is an implicit part of the interface presented to the Media Access sublayer is the round-trip propagation time of the physical medium. Its value represents the maximum time required for a signal to propagate from one end of the network to the other, and for a collision to propagate back. The round-trip propagation time is primarily (but not entirely) a function of the physical size of the network. The round-trip propagation time of the Physical Layer is defined in section 2.14 for a selection of physical media.

2.14 IEEE 802 CSMA/CD: SPECIFIC IMPLEMENTATIONS

To provide total compatibility at all levels of the standard, it is required that each network component implementing the (IEEE 802 CSMA/CD) Media Access Mecha-

nism adheres to these specifications rigidly. The information provided below is a set of tables which provide design parameters for specific implementations of this (IEEE 802) access method. Variations from these values results in a system implementation which violates the standard even though functionally, over a limited set of operating conditions, a particular implementation might work.

The chart below provides recommended values for parameters defined in sections 2.12 and 2.13 for a 10 Mbit per second implementation of an (IEEE 802 CSMA/CD Media) access mechanism. The primary assumptions are that the physical medium is a baseband coaxial cable with properties given in the Media section of the standard.

(Parameters)	(Values)
SlotTime	512 bit times
InterFrameGap	9.6 microsec
AttemptLimit	16
BackOffLimit	10
JamSize	32 bits
MaxFrameSize	1518 octets
MinFrameSize	512 bits
Station Address (DA or SA)	48 bits

CAUTION

Any deviation from the above values specified for a 10 Mbps system may adversely affect proper operation of the LAN.

2.15 CSMA/CD - STATE DIAGRAMS

The following provides a generalized state machine description of the CSMA/CD procedures for media access control. It is supportive of the formal procedures defined in section 2.12. It is assumed that the reader is familiar with those formal descriptions.

The state diagrams provided are descriptive rather than definitional, the formal statements of sections 2.12 through 2.13.2 provide the definitive specifications.

The CSMA/CD Media Access control consists of two components. One component is the transmit component. The other is the receive component. These components operate concurrently and independently.

The transmit component shown in Figure 2-19 (the states of which are listed in Table 2-3) is responsible for handling all events that affect the transmission of a frame onto the media.

2.15.1 Transmit Component Event Descriptions

- Initialize - This event is generated by management to start up the component.
- Data Request - This event is generated by the LLC sublayer. It indicates there is a PDU to be transmitted.
- Carrier On - This event indicates that the physical layer has detected a change in carrier sense from no carrier to carrier.
- Carrier Off - This event indicates that the physical layer has detected a change in the state of carrier sense from carrier to no carrier.
- Preamble Done AND Collision Detect Up - This event indicates that the physical layer has detected a collision with the frame being transmitted and the transmission of the preamble sequence is completed.
- Delay.Timeout - This event indicates that the interframe time delay has completed.
- Backoff.Timeout - This event indicates that the time period for backing off has completed.

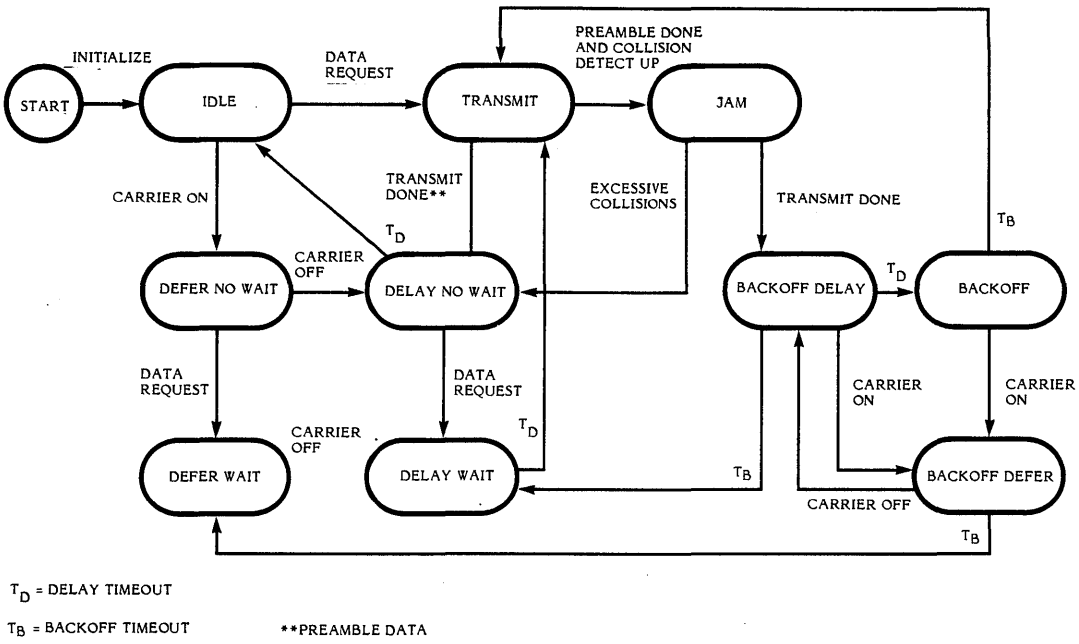


Figure 2-19. Transmit Component State Diagram

- Transmit.Done - The bit transmitter has transmitted all of the bits in the transmit buffer specified by the transmit buffersize (which includes preamble and data).

- Excessive Collisions - The bit transmitter has transmitted all of the bits in the transmit buffer specified by the transmit buffersize, and the attempt count is equal to the maximum transmit attempt count allowed.

2.15.2 Transmit Component Action Descriptions

- Construct Frame - This action encapsulates the data field with the Preamble, SFD, DA, SA, Length, PAD and FCS fields.

- Start Frame Transmission - This action initiates bit transmission of the frame.

- Start Jam Transmission - This action causes the bit transmitter to transmit the bits of the jam pattern.

- Indicate Successful Transmission - This action reports that the transmission was successful.

- Indicate Transmit Failure - This action reports the failure of transmission and the reason.

- Increment Attempt Count - This action increments the counter used to record the number of attempts made to transmit the same frame.

- Reset Attempt Count - This action initializes the attempt count to 0.

- Start Backoff Timer - This action computes the random backoff delay time and sets the backoff timer to that time.

- Start Delay Timer - This action sets the delay timer to the interframe gap time.

Table 2-3. Transmit Component State Transition

CURRENT STATE	EVENT	ACTION	NEXT STATE
0. Start	Initialize	-Perform Initialization	Idle
1. Idle	Data Request	-Construct Frame -Start Frame Transmission	Transmit
	Carrier On	-No Action	Defer No Wait
2. Transmit	Preamble Done AND Collision Detect Up	-Start Jam Transmission -Increment Attempt Count	Jam
	Transmit Done	-Start Delay Timer -Reset Attempt Count -Indicate successful Transmission	Delay No Wait
3. Jam	Transmit Done	-Start Delay Timer -Start Backoff Timer	Backoff Delay
	Excessive Collisions	-Start Delay Timer -Indicate Transmit Excessive Collisions	Delay No Wait
4. Backoff	Carrier On Backoff Timeout	-No Action -Start Frame Transmission	Backoff Defer Tansmit
5. Backoff Defer	Carrier Off Backoff Timeout	-Start Delay Timer -No Action	Backoff Delay Defer Wait
6. Backoff Delay	Carrier On Delay Timeout	-Stop Delay Timer -No Action	Backoff Defer Backoff
	Backoff Timeout	-No Action	Delay Wait
7. Defer No Wait	Data Request Carrier Off	-Construct Frame -Start Delay Timer	Defer Wait Delay No Wait
8. Delay No Wait	Data Request Delay Timeout	-Construct Frame -No Action	Delay Wait Idle
9. Defer Wait	Carrier Off	-Start Delay Timer	Delay Wait
10. Delay Wait	Delay Timeout	-Start Frame Transmission	Transmit

- Stop Delay Timer - This action turns the delay timer off.

- Perform Initialization - This action turns all timers off and ensures that carrier is considered off and collision detect down. All counters are reset. Any implementation specific variables are intialized.

2.15.3 Transmit Component State Descriptions

- Start - The transmit component has not been initialized by management.

- Idle - The transmit component is not transmitting any data nor is it in a state where it is prevented from transmitting data.

- Transmit - The transmit component is actively transmitting bits onto the medium.

- Jam - The transmit component is actively transmitting jam bits onto the medium.

- Backoff - The transmit component is waiting for its random backoff delay to expire before attempting to retransmit a frame.

- Backoff Defer - The transmit component is waiting for both the medium to become available and for its backoff time delay to expire before attempting to retransmit a frame.

- Backoff Delay - The transmit component is waiting for the interframe gap and the backoff delays to expire before attempting to retransmit a frame.

- Defer No Wait - The transmit component has no frame to transmit and it cannot transmit one if it gets one because the medium is busy.

- Delay No Wait - The transmit component has no frame to transmit and it could not if it had one because it is waiting for the interframe gap time to expire.

- Defer Wait - The transmit component is waiting for the medium to become free before attempting to transmit or retransmit the frame.

- Delay Wait - The transmit component is waiting for the interframe gap time to expire before attempting to transmit or retransmit the frame.

The receive component is responsible for handling all events that affect the reception of a frame from the media. The corresponding state diagram is shown in Figure 2-20 and the states are listed in Table 2-4.

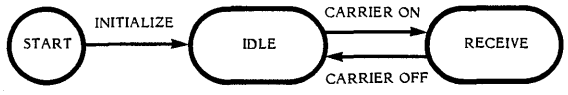


Figure 2-20. Receive Component State Diagram

Table 2-4. Receive Component State Transition

CURRENT STATE	EVENT	ACTION	NEXT STATE
0. Start	Initialize	-Perform Initialization	Idle
1. Idle	Carrier On	-Start Receiving	Receive
2. Receive	Carrier Off	-Process Frame Received	Idle

2.15.4 Receive Component Event Descriptions

- o Initialize - This event is generated by management to start up the component.

- o Carrier On - This event indicates that the physical layer has detected a change in carrier sense from no carrier to carrier.

- Carrier Off - This event indicates that the physical layer has detected a change in the state of carrier sense from carrier to no carrier.

2.15.5 Receive Component Action Descriptions

- Perform Initialization - This action turns all timers off and ensures that carrier is considered off and collision detect down. All counters are reset. Any implementation specific variables are initialized.

- Start Receiving - This action begins the processes of accepting bits and appending them to the buffer used to contain the frame.

- Process Frame Received - If the frame is not addressed to this station, then ignore the frame. Otherwise, check the frame for errors. If there are no errors, pass frame up to the LLC sublayer indicating no error. Otherwise, pass the frame to the LLC sublayer indicating the error.

2.15.6 Receive Component State Descriptions

- Start - The receive component has not been initialized by management.

- Idle - The receive component is not actively receiving bits of data from the line.

- Receive - The receive component is receiving bits of data from the line.

CHAPTER 3
NETWORK LAYER

3.1 INTRODUCTION

The Network Layer Architecture described in this section is specific to release number 1 of iNA 960.

It is intended that the architecture specified will provide an Internet service interface that looks like the ISO internet standard. The goal is to make the upper layer software transparent to future adoption of the ISO network standard. The specified architecture follows the most current information available for the network layer standard process (see Information Processing Systems - Data Communications - Connectionless Internetwork Protocol Specifications, ISO TC97/SC 6, 97.6.32.4, January 1983).

3.1.1 Overview

The Release 1 version of iNA 960 provides a minimal network protocol. It is characterized by:

- Implementation of the Zero-Length protocol; no internetwork routing is employed. The network layer provides datagram delivery in a single LAN.

- Provision of a full Internet Address at the network service interface. The transport layer must use a full internet address for local delivery.

- Provision of buffer management functions for data links, the transport layer, and the session layer.

The network layer follows the ECMA three-sublayer model; see section 3.2.1.

Internet Addressing is not a critical issue in the Release 1 of iNA 960 because the internet address is not included in the protocol. However, the model and the format of the internet address is critical because it is used in Release 1 of iNA 960 network layer interface. The format follows the ISO internet address model. Section 3.2.2 describes the interpretation of this model used in Release 1 of iNA 960.

The function of Buffer Management is to maintain balanced buffer allocation between transport and data links. Buffer management also provides a means for throughput adjustment, and congestion controls.

3.1.2 Related Documents

The following documents are sources of information pertinent to the subject of this chapter:

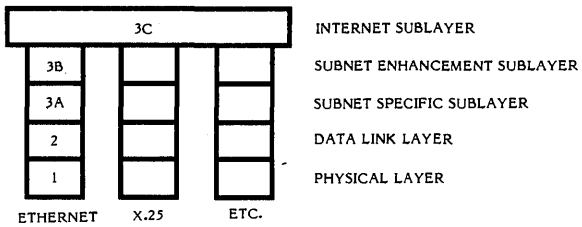
- Information Processing Systems - Data Communications - Connectionless Internetwork Protocol Specifications, ISO TC97/SC6, 97.6.32.4, January 1983.
- Network Layer Principles, ECMA TR/13, September 1982.
- Working Paper on Internet Protocol, ECMA, October 1982.

3.2 NETWORK LAYER ARCHITECTURE

In Release 1 of iNA 960, the network layer provides for datagram services within a single, non-interconnected, Ethernet LAN. Although it does not provide any internetwork routing, its interface supports full "internet address". The architecture follows the ECMA network model (see Network Layer Principle, ECMA TR13, September 1982).

3.2.1 Network Layer Model

The ECMA network model consists of three sublayers: the Internet Sublayer (3C), the Subnet Enhancement Sublayer (3B), and the Subnet Specific Sublayer (3A). The following illustrates the sublayers and their relationships to the "subnets".



A "subnet" is a set of nodes that shares a common addressing convention and can be considered an independent, self-contained entity. In the above illustration, Ethernet, X.25, etc., are different subnets. Release 1 of iNA 960, supports only the Ethernet subnet.

In the ECMA model, the Internet Sublayer performs routing between the various subnets associated with the 3A sublayer. The Internet Sublayer does this by imposing a uniform "internet address" on all subnet nodes. In Release 1 of iNA 960, all of the network services interfaces are defined in terms of internet addressing.

The Subnet Enhancement Sublayer provides any subnet specific enhancement (or de-enhancement) protocols needed to achieve a subnet-service level required by the Internet Sublayer. In Release 1 of iNA 960, this sublayer is null.

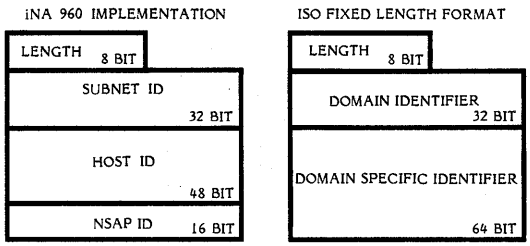
The Subnet Specific Sublayer is part of the subnet access protocols. For Ethernet subnet, this sublayer is null.

3.2.2 Internet Addressing

The Internet Address used follows the "Fixed Length Format" internet address defined by ISO (see Information Processing Systems - Data Communications - Connectionless Internetwork Protocol Specifications, ISO TC 97/SC 6, 97.6.32.4, January 1983). In this format, the internet address consists of a 32-bit "domain

identifier" and a 64-bit "domain specific identifier". The domain identifier uniquely identifies each domain, or subnet. The domain identifier is provided by ISO. The format of the domain specific identifier is an issue within the particular domain, and is not defined by ISO at the internet level.

The illustration below shows how Release 1 of iNA 960 adopts the ISO internet address format.



In Release 1 of iNA 960, the internet address consists of a 32-bit subnet identifier (ID), a 48-bit Ethernet host ID, and a 16-bit NSAP ID. The Ethernet ID and the NSAP ID form the domain specific identifier, which is an issue determined within the Ethernet community.

The Length field is maintained so that this internet address format can fit well into the future "Variable Length Format" internet address scheme.

The internet address is employed in Release 1 of iNA 960 at the network service interface point. Release 1 of iNA 960 does not employ any internet address in the network protocol. (For a discussion of the network protocol, see Section 3.2.4.)

In Release 1 of iNA 960, the internet address must contain the following values:

Length = 0CH
Subnet ID = 1
Subnet Specific = 48-bit Ethernet Host ID
NSAP ID = 1

A request with a Subnet ID other than 1 will receive a "cannot reach" error response.

3.2.3 NSAP ID

Release 1 of iNA 960 supports only one Network Service Access Point on each node. This NSAP is to be used by the transport layer only. The transport must use this single NSAP for virtual circuit service and the datagram service. No NSAP is to be used by the Network Management Layer or any external users. For any datagram services, the external users or the Network Management Layer must use the transport datagram service.

As part of the internet address, the NSAP ID in Release 1 of iNA 960 is only for network service interface issue. It is not a protocol issue.

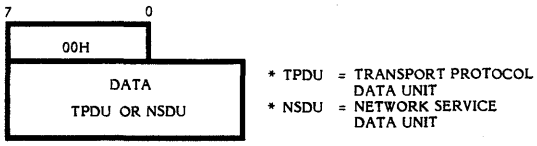
NOTE

The NSAP ID described here differs from the "NSAP Address" (NSAPA) described in some ECMA documents. NSAPA refers to the entire network address (or internet address). The NSAP ID is like any other "XSAP ID," which are identifiers for multiplexing within one service access point.

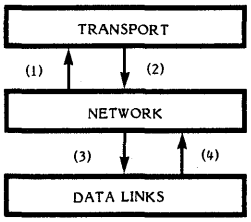
3.2.4 Protocol of the Network Layer

No internetwork protocol is employed in Release 1 of iNA 960. The "Zero Length Protocol" defined by ISO fits this (no internetworking is performed).

Zero Length Protocol has the following format:



A one-byte zero length indicator is the only network protocol control information, indicating that no internetwork protocol is employed.

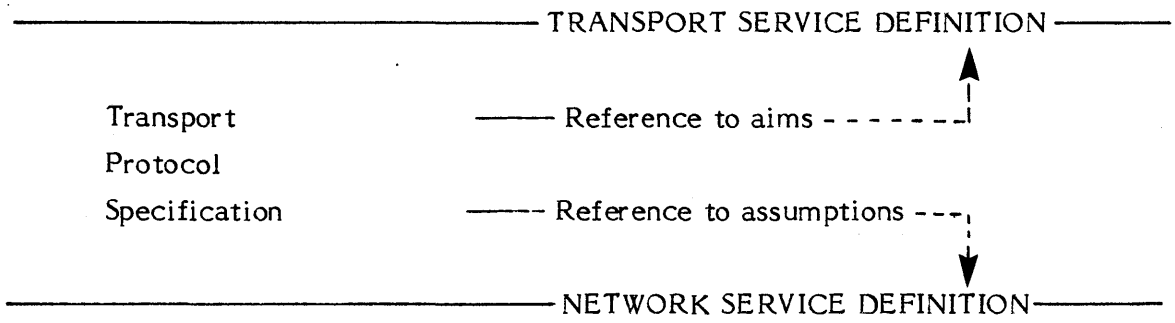


CHAPTER 4
TRANSPORT LAYER (CONNECTION ORIENTED)

4.1 INTRODUCTION

The Transport Protocol specified in this section follows the ISO Standard; one of a set of International Standards produced to facilitate the interconnection of computer systems. The set of standards covers the services and protocols required to achieve such interconnection.

The Transport Protocol Standard is positioned with respect to other related standards by the layers defined in the Reference Model for Open Systems Interconnection (ISO 7498). It is most closely related to, and lies within the field of application of the Transport Service Standard (DP 8072). It also uses and makes reference to the Network Service Standard (DP bbbb, see section 4.2.3), whose provisions it assumes in order to accomplish the transport protocol's aims. The interrelationship of these standards is depicted in the following illustration:



The International Standard specifies a common encoding and a number of classes of transport protocol procedures to be used with different network qualities of service.

The Transport Protocol is simple but general enough to cater for the total range of Network Service qualities possible, without restricting future extensions.

The protocol is structured to give rise to classes of protocol which are designed to minimize possible incompatibilities and implementation costs.

The classes are selectable with respect to the Transport and Network Services in providing the required quality of service for the interconnection of two session entities (note that each class provides a different set of functions for enhancement of service qualities).

The ISO Protocol Standard is concerned with optimization of network tariffs and the following qualities of service:

- Different throughput rates;
- Different error rates;
- Integrity of data requirements;
- Reliability requirements.

The primary aim of the ISO Standard and of this chapter is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer entities at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

- As a guide for implementors and designers;
- For use in the testing and procurement of equipment;
- As part of an agreement for the admittance of systems into the open systems environment.

The following exerpt (NOTE) from ISO document DP8073 is included here mainly to indicate the reasons for the many options and variations in the specification of the transport layer architecture.

NOTE

It is expected that the initial users of the International Standard will be designers and implementors of equipment; and, the International Standard contains, in notes or in annexes, guidance on the implementation of the procedures defined in the standard.

It should be noted that, as the number of valid protocol sequences is very large, it is not possible with current technology to verify that an implementation will operate the protocol defined in this International Standard correctly under all circumstances. It is possible by means of testing to establish confidence that an implementation correctly operates the protocol in a representative sample of circumstances. It is, however, intended that this International Standard can be used in circumstances where two implementations fail to communicate in order to determine whether one or both have failed to operate the protocol correctly.

It has not been possible at the present time to prepare a product standard containing a set of objective tests but this International Standard contains a section on conformance of equipment claiming to implement the procedures in this International Standard. Attention is drawn to the fact that the standard does not contain any tests to demonstrate this conformance and cannot, therefore, be considered as a complete product standard.

The variations and options available within this International Standard are essential to enable a Transport Service to be provided for a wide variety of applications over a variety of network qualities. Thus, a minimally conforming implementation will not be suitable for use in all possible circumstances. It is important, therefore, to qualify all references to this International Standard with statements of the options provided or required or with statements of the intended purpose of provision or use.

4.2 SCOPE AND FIELD OF APPLICATION

4.2.1 Scope

- This chapter specifies five classes of procedure:

  Class 0--Simple class,
  Class 1--Basic error recovery class,
  Class 2--Multiplexing class,
  Class 3--Error recovery and multiplexing class,
  Class 4--Error detection and recovery class,

  for the connection oriented transfer of data and control information from one transport entity to a peer transport entity;

- the means of negotiating the class of procedures to be used by the transport entities;

- the structure and encoding of the transport protocol data units used for the transfer of data and control information.

The R1 version of iNA 960 is class 4 (only), but all classes are discussed for completeness and contrast.

4.2.2 Procedures and Their Application

The procedures are defined in terms of:

- the interactions between peer transport entities through the exchange of transport protocol data units;

- the interactions between a transport entity and the transport service user in the same system through the exchange of transport service primitives;

- the interactions between a transport entity and the network service provider through the exchange of network service primitives.

The procedures are defined in the main text of this chapter supplemented by state tables at the end of the chapter.

These procedures are applicable to instances of communication between systems which support the Transport Layer (OSI Reference Model) and which wish to interconnect in an open systems environment.

This section also specifies ISO standard conformance for systems implementing these procedures. It does not contain tests which can be used to demonstrate this conformance.

4.2.3 References

The following documents contain information relevant to the content of this chapter:

ISO 7498 Information processing systems - Open systems interconnection - Basic Reference Model.

DP 8072 Information processing systems - Open systems interconnection - Transport service definition.

DP bbbb Information processing systems - Open systems interconnection - Connection-oriented network service definition (ISO/TC97/SC6 N2610).

4.2.4 Definitions

The ISO Standard is based on the concepts developed in the Reference Model for Open Systems Interconnection (DIS 7498) and makes use of the following terms defined in that standard:

- concatenation and separation;
- segmenting and reassembling;
- multiplexing and demultiplexing;
- splitting and recombining;
- flow control.

For the purpose of this chapter (and the ISO Standard), the following definitions apply:

equipment: Hardware or software or a combination of both; it need not be physically distinct within a computer system.

transport service user: An abstract representation of the totality of those entities within a single system that make use of the transport service.

network service provider: An abstract machine that models the totality of the entities providing the network service, as viewed by a transport entity.

local matter: A decision made by a system concerning its behavior in the Transport Layer that is not subject to the requirements of this protocol.

initiator: A transport entity that initiates a Connection Request Transport Protocol Data Unit (CR TPDU).

responder: A transport entity with whom an initiator wishes to establish a transport connection.

NOTE

Initiator and responder are defined with respect to a single transport connection. A transport entity can be both an initiator and responder simultaneously.

sending transport entity: A transport entity that sends a given Transport Protocol Data Unit (TPDU).

receiving transport entity: A transport entity that receives a given Transport Protocol Data Unit (TPDU).

preferred class: The protocol class that the initiator indicates in a CR TPDU as its first choice for use over the transport connection.

alternative class: A protocol class that the initiator indicates in a CR TPDU as an alternative choice for use over the transport connection.

proposed class: A preferred class or an alternative class.

selected class: The protocol class that the responder indicates in a Connection Confirm Transport Protocol Data Unit (CC TPDU) that it has chosen for use over the transport connection.

proposed parameter: The value for a parameter that the initiator indicates in a CR TPDU that it wishes to use over the transport connection.

selected parameter: The value for a parameter that the responder indicates in a CC TPDU that it has chosen for use over the transport connection.

error indication: An N-RESET indication ("N" = maximum number of transmissions), or an N-DISCONNECT indication with a reason code indicating an error, that a transport entity receives from the NS-provider (NS = Network Service).

invalid TPDU: A TPDU that does not comply with the requirements of the International Standard for structure and encoding.

protocol error: A TPDU whose use does not comply with the procedures for the class.

sequence number: There are two definitions for this, depending on context:

a) The number in the DT TPDU number (TPDU-NR) field of a Data TPDU (DT TPDU) that indicates the order in which the DT TPDU was transmitted by a transport entity.

b) The number in the sequence number response (YR-TU-NR) field of a Data Acknowledge (AK) or Reject (RJ) TPDU that indicates the sequence number of the next DT TPDU expected to be received by a transport entity.

transmit window: The set of consecutive sequence numbers which a transport entity has been authorized by its peer entity to send at a given time on a given transport connection.

lower window edge: The lowest sequence number in a transmit window.

upper window edge: The sequence number which is one greater than the highest sequence number in the transmit window.

upper window edge allocated to the peer entity: The value that a transport entity communicates to its peer entity to be interpreted as its new upper window edge.

closed window: A transmit window that contains no sequence number.

window information: Information contained in a TPDU relating to the upper and lower window edges.

frozen reference: A reference that is not available for assignment to a connection because of the requirements set forth in section 4.4.19.

unassigned reference: A reference that is neither currently in use for identifying a transport connection nor is in a frozen state.

transparent (data): TS-user (TS = Transport Service) data that is transferred intact between transport entities and which is unavailable for use by the transport entities.

owner (of a network connection): The transport entity that issued the N-CONNECT request leading to the creation of that network connection.

retained TPDU: A TPDU that is subject to the retransmission procedure or retention-until-acknowledgement procedure and is available for possible retransmission.

4.2.5 Symbols and Abbreviations

- Data units

TPDU	Transport protocol data unit
TSDU	Transport service data unit
NSDU	Network service data unit

- Types of transport protocol data units

CR TPDU	Connection request TPDU
CC TPDU	Connection confirm TPDU
DR TPDU	Disconnect request TPDU
DC TPDU	Disconnect confirm TPDU
DT TPDU	Data TPDU
ED TPDU	Expedited data TPDU
AK TPDU	Data acknowledge TPDU
EA TPDU	Expedited acknowledge TPDU
RJ TPDU	Reject TPDU
ER TPDU	Error TPDU

- TPDU fields

LI	Length indicator (field)
CDT	Credit (field)
TSAP-ID	Transport service access point identifier (field)
DST-REF	Destination reference (field)

SRC-REF	Source reference (field)
EOT	End of TSDU mark
TPDU-NR	DT TPDU number (field)
ED-TPDU-NR	ED TPDU number (field)
YR-TU-NR	Sequence number response (field)

- Timer variables

T1	Elapse time between retransmissions
N	The maximum number of transmissions
L	Bound on reference
I	Inactivity timer
W	Window time
TTR	Time to try reassignment after failure
TWR	Time to wait for reassignment
TS1	Supervisory timer for connection establishment
TS2	Supervisory timer for connection release

- Miscellaneous

TS-user	Transport service user
TSAP	Transport service access point
NS-provider	Network service provider
NSAP	Network service access point

4.3 OVERVIEW OF THE TRANSPORT PROTOCOL

The following subsections provide an overview of the transport layer in terms of service provided, network layer service assumed, functions performed, classes and options and the standard model.

4.3.1 Service Provided by the Transport Layer

The protocol specified in this chapter supports the transport service defined in DP 8072.

Information is transferred to and from the TS-user in the transport service primitives listed in Table 4-1.

Table 4-1. Transport Service Primitives

Primitive		Parameters
T-CONNECT	request indication	Called Address, Calling Address Expedited Data option, Quality of Service, TS User-Data
T-CONNECT	response confirm	Responding Address, Quality of Service, Expedited Data option, TS User-Data.
T-DATA	request indication	TS User-Data
T-EXPEDITED DATA	request	TS User-Data
T-DISCONNECT	request	TS User-Data
T-DISCONNECT	indication	Disconnect reason, TS UserData

4.3.2 Service Assumed From the Network Layer

The protocol specified in this International Standard assumes the user of the network service defined in DP bbbb (see section 4.2.3).

Information is transferred to and from the NS-provider in the network service primitives listed in Table 4-2.

4.3.3 Functions of the Transport Layer

The functions in the Transport Layer are those necessary to bridge the gap between the services available from the Network Layer and those to be offered to the TS-users.

Table 4-2. Network Service Primitives

Primitives		X/Y	Parameters	X/Y/Z
N-CONNECT	request	X	Called Address,	X
	indication	X	Calling Address,	X
	response	X	NS User-Data,	Z
	confirm	X	QOS parameter set.	X
N-Data	request	X	NS User-Data,	X
	indication	X	Confirmat. request	Y
N-DATA ACKNOWLEDGE				
	request	Y		
	indication	y		
N-EXPEDITED DATA				
	request	Y	NS-User-Data	Y
	indication	Y		
N-RESET	request	X		
	indication	X		
	response	X		
	confirm	X		
N-DISCONNECT	request	X	NS-User-Data	Z
	indication	X		

NOTE: X The Transport Protocol assumes that this facility is provided in all networks.
Y The Transport Protocol assumes that this facility is provided in some networks and a mechanism is provided to use the facility optionally.
Z The Transport Protocol does not use this parameter.

Note that the parameters listed in this table are not exhaustive.

The functions in the Transport Layer are concerned with the enhancement of quality of service, including all aspects of cost optimization.

These functions are grouped below into those used at all times during a transport connection and those concerned with connection establishment, data transfer and release.

NOTE

This manual and DP 8073 do not include the following functions which are under consideration for inclusion in future editions of this standard:

- o encryption;
- o accounting mechanisms;
- o status exchanges and monitoring of QOS;
- o blocking;
- o temporary release of network connections.

The following functions, if appropriate, are used at all times during a transport connection:

a) Transmission of TPDUs (see sections 4.4.2 and 4.4.9);

b) Multiplexing and demultiplexing (see section 4.4.15), a function used to share a single network connection between two or more transport connections;

c) Error detection (see sections 4.4.10, 4.4.13, and 4.4.17), a function used to detect the loss, corruption, duplication, misordering or misdelivery of TPDUs;

d) Error recovery (see sections 4.4.12, 4.4.14, 4.4.18, 4.4.19, 4.4.20, 4.4.21 and 4.4.22), a function used to recover from detected and signalled errors.

The purpose of connection establishment is to establish a transport connection between two TS-users. The functions of the transport layer during this phase must match the TS-users' requested quality of service with the following services provided by the network layer (in each case, for more detail, see section 4.4.5):

a) Select network service which best matches the requirements of the TS-user taking into account charges for various services;

b) Decide whether to multiplex multiple transport connections onto a single network connection;

c) Establish the optimum TPDU size;

d) Select the functions that will be operational upon entering the data transfer phase;

e) Map transport addresses onto network addresses;

f) Provide a means to distinguish between two different transport connections;

g) Transport of TS-user data.

The purpose of data transfer is to permit duplex transmission of TSDUs between the two TS-users connected by the transport connection. This purpose is achieved by means of two-way simultaneous communication in the Transport Protocol and by the following functions, some of which are used or not used in accordance with the result of the selection performed in connection establishment:

a) Concatenation and separation (see section 4.4.4), a function used to collect several TPDUs into a single NSDU at the sending transport entity and to separate the TPDUs at the receiving transport entity;

b) Segmenting and reassembling (see section 4.4.3), a function used to split a single data TSDU into multiple TPDUs at the sending transport entity and to reassemble them into their original format at the receiving transport entity;

c) Splitting and recombining (see section 4.4.23), a function allowing the simultaneous use of two or more network connections to support the same transport connection;

d) Flow control (see section 4.4.16), a function used to regulate the flow of TPDUs between two transport entities on one transport connection;

e) Transport connection identification, a means to uniquely identify a transport connection between a pair of transport entities supporting the connection during the lifetime of the transport connection;

f) Expedited data (see section 4.4.11), a function used to bypass the flow control of normal data TPDU. Expedited data TPDU flow is controlled by separate flow control;

g) TSDU delimiting (see section 4.4.3), a function used to determine the beginning and ending of a TSDU.

The purpose of release (see sections 4.4.7 and 4.4.8) is to provide disconnection of the transport connection, regardless of the current activity.

4.3.4 Classes and Options

The functions of the Transport Layer are organized into classes and options.

A class defines a set of functions. Options define functions which may or may not be used within a class.

Five classes of protocol are defined:

Class 0: Simple Class;

Class 1: Basic Error Recovery Class;

Class 2: Multiplexing Class;

Class 3: Error Recovery and Multiplexing Class;

Class 4: Error Detection and Recovery Class.

NOTE

With the exception of Classes 0 and 1, transport connections of different classes may be multiplexed together onto the same network connection.

The use of classes and options is negotiated during connection establishment. The choice made by the transport entities depends upon:

- the TS-users' requirements expressed via T-CONNECT service primitives,
- the quality of the available network services,
- the user required service versus cost ratio acceptable to the TS-user.

The following list classifies network services in terms of quality with respect to error behavior in relation to user requirements; its main purpose is to provide a basis for the decision regarding which class of transport protocol should be used on top of a given network connection:

Type A. Network connections with acceptable residual error rate (for example not signalled by 'clear' or 'reset') and acceptable rate of signalled failures.

Type B. Network connections with acceptable residual error rate (for example not signalled by 'clear' or 'reset') but unacceptable rate of signalled failures.

Type C. Network connections with residual error rate not acceptable to the TS-user.

It is assumed that each transport entity is aware of the quality of service provided by particular network connections.

The following are the characteristics of the several classes:

Class 0 provides the simplest type of transport connection and is fully compatible with the CCITT recommendation S.70 for teletex terminals. Class 0 has been designed to be used with type A network connections.

Class 1 provides a basic transport connection with minimal overheads. The main purpose of class 1 is to recover from network signalled errors (network disconnect or reset).

Selection of this class is usually based on reliability criteria. Class 1 has been designed to be used with type B network connections.

Class 2 provides a way to multiplex several transport connections onto a single network connection. This class has been designed to be used with type A network connections. In this class, explicit use of flow control is optional.

The objective explicit use of flow control is to help avoid congestion at end-points and on the network connection. Typically it is used when traffic is heavy and continuous, or when there is intensive multiplexing. Use of flow control can optimize response times and resource utilization.

The objective of the "Non-use of explicit flow control" is to provide a basic transport connection with minimal overheads suitable when independence of transport and network connection lifetime is desirable. This option would typically be used for unsophisticated terminals, and when no multiplexing onto network connections is required. Expedited data is never available.

Class 3 provides the characteristics of Class 2 plus the ability to mask errors indicated by the network. Selection of this class is usually based upon reliability criteria. Class 3 is designed to be used with type B network connections.

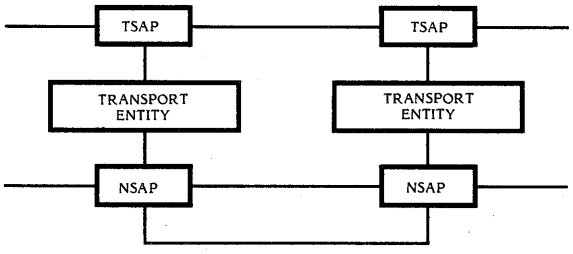
Class 4 provides the characteristics of Class 3 plus the detection of errors which occur as a result of the low grade of service available from the NS-provider. The kinds of errors to be detected include: TPDU loss, TPDU delivery out of sequence, TPDU duplication and TPDU corruption. These errors may affect control TPDUs as

well as data TPDUs. Class 4 is designed to be used with type C network connections.

4.3.5 Model of the Transport Layer

A transport entity communicates with its TS-users through one or more TSAPs by means of the service primitives as defined by the transport service definition DP 8072. Service primitives cause or are the result of transport protocol data unit exchanges between the peer transport entities supporting a transport connection. These protocol exchanges are effected using the services of the Network Layer as defined by the Network Service Definition DP bbbb (see 4.2.3) through one or more NSAPs.

Transport connection endpoints are identified in end systems by an internal, implementation dependent, mechanism so that the TS-user and the transport entity can refer to each transport connection (see Figure 4-1).



NOTE

For purpose of illustration, this figure shows only one TSAP and one NSAP for each transport entity. In certain instances, more than one TSAP and/or more than one NSAP may be associated with a particular transport entity.

Figure 4-1. Model of the Transport Layer

4.4 ELEMENTS OF PROCEDURE

The following sections contain elements of procedure used in the specification of protocol classes discussed in sections 4.4 through 4.10. These elements are not meaningful on their own.

The procedures define the transfer of TPDUs whose structure and coding is specified in section 4.11. Transport entities must accept and respond to any TPDU received in a valid NSDU and may issue TPDUs initiating specific elements of procedure specified below.

NOTE

Where NSDUs and TPDUs and parameters used are not significant for a particular element of procedure, they are not included.

4.4.1 Assignment to Network Connection

The procedure is used in all classes to assign transport connections to network connections.

The procedure makes use of the following network service primitives:

a) N-CONNECT;

b) N-DISCONNECT.

Each transport connection must be assigned to a network connection. The initiator may assign the transport connection to an existing network connection of which it is the owner or to a new network connection (see Note 1 below) which it creates for this purpose.

The initiator must not assign or reassign the transport connection to an existing network connection if the protocol class(es) proposed or the class in use for the transport connection are incompatible with the current usage of the network connection with respect to multiplexing (see Note 2 below).

During the resynchronization (see section 4.4.14) and reassignment after failure (see section 4.4.12) procedures, a transport entity may reassign a transport connection to another network connection joining the same NSAPs, provided that it is the owner of the network connection and that the transport connection is assigned to only one network connection at any given time.

During the splitting procedure (see section 4.4.23), a transport entity may assign a transport connection to any additional network connection joining the same NSAPs, provided that it is the owner of the network connection and that multiplexing is possible on the network connection.

The responding transport entity becomes aware of the assignment when it receives:

- A CR TPDU during the connection establishment procedure (see section 4.4.5), or
- An RJ TPDU or a retransmitted CR or DR TPDU during the resynchronization (see section 4.4.14) and reassignment after failure (see section 4.4.12) procedures, or
- Any TPDU when splitting (see section 4.4.23) is used.

NOTE 1

When a new network connection is created, the quality of service requested is a local matter, although it will normally be related to the requirements of transport connection(s) expected to be assigned to it.

NOTE 2

An existing network connection may also not be suitable if, for example, the quality of service requested for the transport connection cannot be attained by using or enhancing the network connection.

NOTE 3

A network connection with no transport connection(s) assigned to it, may be available

after initial establishment, or because all of the transport connections previously assigned to it have been released. It is recommended that only the owner of such a network connection should release it. Furthermore, it is recommended that it not be released immediately after the transmission of the final TPDU of a transport connection - either a DR in response to CR or a DC in response to DR. An appropriate delay will allow the TPDU concerned to reach the other transport entity allowing the freeing of any resources associated with the transport connection concerned.

NOTE 4

After the failure of a network connection, transport connections which were previously multiplexed together may be assigned to different network connections, and vice versa.

4.4.2 Transport Protocol Data Unit (TPDU) Transfer

The TPDU transfer procedure is used in all classes to convey transport protocol data units in user data fields of network service primitives. The procedure uses the following network service primitives:

a) N-DATA;

b) N-EXPEDITED DATA.

The transport protocol data units (TPDUs) defined for the protocol are listed in section 4.16.

When the network expedited variant has been selected for class 1, the transport entities must transmit and receive ED and EA TPDUs as NS-user data parameters of N-EXPEDITED DATA primitives. In all other cases, transport entities must transmit and receive TPDUs as NS-user data parameters of N-DATA primitives.

NOTE

Concatenation of TPDUs may also be permitted (see section 4.4.4).

4.4.3 Segmenting and Reassembling

The segmenting and reassembling procedure is used in all classes to map TSDUs onto TPDUs. The procedure makes use of the DT TPDUs and the End of TSDU parameter.

A transport entity may map a TSDU onto an ordered sequence of one or more DT TPDUs. This sequence must not be interrupted by other DT TPDUs on the same transport connection.

All DT TPDUs, except the last DT TPDU, in a sequence greater than one must have a length of data greater than zero. Violation of this rule is a protocol error.

NOTE

The end of TSDU parameter of a DT TPDU indicates whether or not there are subsequent DT TPDUs in the sequence. Also, there is no requirement that the DT TPDUs be of the maximum length selected during connection establishment.

4.4.4 Concatenation and Separation

The procedure for concatenation and separation is used in classes 1, 2, 3 and 4 to convey multiple TPDUs in one NSDU.

A transport entity may concatenate TPDUs from the same or different transport connections. The set of concatenated TPDUs may contain:

- any number of TPDUs that do not contain user data, provided that these TPDUs come from different transport connections;
- no more than one TPDU containing user data; if this TPDU is present, it must be placed last in the set of concatenated TPDUs.

NOTE

The TPDUs within a concatenated set may be distinguished by means of the length indicator parameter. Also, the end of a TPDU containing data is indicated by the termination of the NSDU.

4.4.5 Connection Establishment

The procedure for connection establishment is used in all classes to create a new transport connection.

The procedure uses the N-DATA network service primitive and the following TPDUs and parameters:

Primitive	Parameters
CR TPDU	CDT; DST-REF (set to zero); SRC-REF; CLASS and OPTIONS (preferred; i.e. class, use of extended format, non-use of explicit flow control in class 2); calling TSAP-ID; called TSAP-ID; TPDU size (proposed); version number; security parameter; checksum; additional option selection (preferred: use of network expedited in class 1, use of receipt confirmation in class 1); non-use of checksums in class 4, use of transport expedited data transfer service; alternative protocol class; acknowledge time; throughput (proposed); residual error rate (proposed); priority (proposed); transit delay (proposed); reassignment time; user data.
CC TPDU	CDT; DST-REF; SRC-REF; CLASS and OPTIONS (selected); calling TSAP-ID;

Primitive	Parameters
	called TSAP-ID; TPDU size (selected); security parameter; checksum; additional option selection (selected); acknowledge time; throughput (selected); residual error rate (selected); priority (selected); transit delay (selected); user data.

A transport connection is established by means of one transport entity (the initiator) transmitting a Connection Request (CR) TPDU to the other transport entity (the responder), which replies with a Connection Confirm (CC) TPDU. Before sending the CR TPDU, the initiator assigns the transport connection being created to one (or more if the splitting procedure is being used) network connection(s). It is this set of network connections over which the TPDUs are sent. During this exchange, all information and parameters needed for the transport entities to operate must be exchanged or negotiated.

NOTE

It is recommended that the initiator starts a timer TS1 at the time the CR TPDU is sent. This timer should be stopped when the connection is considered as accepted or refused or unsuccessful. If the timer expires, the initiator should release the network connection and, in classes 1, 3 and 4, freeze the reference (see section 4.4.19).

After receiving the CC TPDU for a class which includes the procedure for retention until acknowledgement of TPDUs, the initiator must acknowledge the CC TPDU as defined in Table 4-5 in section 4.4.13.

When the network expedited variant of the expedited data transfer (see section 4.4.11) has been agreed upon (possible class 1 only), the responder must not send an ED TPDU before the CC TPDU is acknowledged.

The following information is exchanged:

References. Each transport entity chooses a reference which is 16 bits long and which is arbitrary except for the following restrictions:

- It must not already be in use or 'frozen' (see section 4.4.18),
- It must not be zero.

Each transport entity is responsible for selecting the reference which the partner is to use. This mechanism is symmetrical. This mechanism also provides identification of the transport connection independent of the network connection. The range of references used for transport connections, in a given transport entity, is a local matter.

Addresses (optional). Addresses indicate the calling and called transport service access points. When either network address unambiguously defines the transport address, this information may be omitted.

Initial credit. This is only relevant for classes which include the explicit flow control function.

User data. This is not available if class 0 is the preferred class (see the following note). Up to 32 octets of data are allowed in other classes.

NOTE

If class 0 is a proposed class, inclusion of user data in the CR TPDU may cause the responding entity to refuse the connection (e.g. if it only supports class 0).

Acknowledgement time. This is used only in class 4.

The following negotiations take place:

Protocol class. The initiator must propose a preferred class and may propose any number of alternative classes which permit a valid response as defined in Table 4-3. The initiator should assume when it sends the CR TPDU that its preferred class will be agreed to, and commence the functions associated with that class, except that if class 0 or class 1 is an alternative class, multiplexing must not commence until a CC TPDU selecting the use of classes 2, 3 or 4 is received. This means, for example, that when a class which includes resynchronization (see section 4.4.14) is preferred, resynchronization will occur if a reset is signalled during connection establishment.

Table 4-3. Valid Responses Corresponding to Preferred and Alternative Classes in the CR TPDU

Preferred Class	0	1	2	3	3 and 1	4	None
0	not valid	not valid	not valid	not valid	not valid	not valid	class 0
1	class 1 or 0	class 1 or 0	not valid	not valid	not valid	not valid	class 1 or0
2	class 2 or 0	not valid	class 2	not valid	not valid	not valid	class 2
3	class 3,2 or 0	class 3, 2,1 or 0	class 3 or 2	class 3 or 2	class 3, 2,1 or 0	not valid	class 3 or 2
4	class 4,2,or 0	class 4 2,1 or 0	class 4 or 2	class 4,3 or 2	class 4 3,2,1or0	class 4 or 2	class 4 or 2

The responder must select one class defined as valid in Table 4-3. It must indicate this in the CC TPDU and invoke the appropriate functions for the class.

If the preferred class is not selected, then on receipt of the CC TPDU, the initiator must adjust its functions accordingly.

TPDU size. The initiator may propose a maximum size for TPDUs, and the responder may accept this value or respond with any value between the proposed value and 128 in the set of values available (see section 4.11).

NOTE

The length of the CR TPDU does not exceed 128 octets (see section 4.10).

Normal or extended format. Either normal or extended is available. When extended is used, this applies to CDT, TPDU-NR, ED-TPDU-NR, YR-TU-NR parameters.

Checksum selection. This defines whether or not TPDUs of the connection are to include a checksum.

Version number. This defines the version of the transport protocol standard used for this connection.

Security parameter. This parameter and its semantics are user defined.

Quality of service parameters. This defines the throughput, transit delay, priority and residual error rate. Only those QOS parameters are reflected which are of global significance.

Non-use of explicit flow control. This is negotiated when class 2 is to be used.

Use of network receipt confirmation and network expedited. This is negotiated when class 1 is to be used.

Reassignment time parameter. This indicates the time span over which the initiator will persist in following the reassignment after failure procedure.

The negotiation rules for the options are such that the initiator may propose either to use or not to use the option. The responder may either accept the proposed choice or select an alternative choice as defined in Table 4-4. During connection establishment, the use of the expedited data parameter field of CR and CC TPDUs allows both TS-users to negotiate the use or non-use of the expedited data transport service as described in the transport service definition.

In class 2, whenever a transport entity requests or agrees to the transport expedited data transfer service or to the use of extended formats, it must also request or agree (respectively) to the use of explicit flow control.

Table 4-4. Negotiation of Options During Connection Establishment

Option	Proposition Made by the Initiator	Possible Selection by the Responder
Transport Expedited data transfer service (Classes 1,2,3,4 only)	Yes No	Yes or No No
Use of receipt confirmation (Class 1 only)	Yes No	Yes or No No
Use of the network expedited variant (Class 1 only)	Yes No	Yes or No No
Non-use of checksum (Class 4 only)	Yes No	Yes or No No
Non-use of explicit flow control (Class 2 only)	Yes No	Yes or No No
Use of extended format (Classes 2,3,4 only)	Yes No	Yes or No No

NOTE: This table defines the procedures for negotiation of options. This negotiation is designed such that if the initiator proposes the mandatory implementation options specified in section 4.12, the responder has to accept use of this option over the transport connection. If the initiator proposes a non-mandatory implementation option, the responder is entitled to select use of the mandatory implementation option for use over the transport connection.

4.4.6 Connection Refusal

The connection refusal procedure is used in all classes when a transport entity refuses a transport connection in response to a CR TPDU.

The procedure makes use of the following TPDUs and parameters:

TPDU	Parameters
DR TPDU	SRC-REF; reason; user data.
ER TPDU	reject code; rejected TPDU parameter.

If a transport connection cannot be accepted, the called transport entity must respond to the CR TPDU with a DR TPDU. The reason must indicate why the connection was not accepted. The source reference field in the DR TPDU is set to zero to indicate an unassigned reference.

If a DR TPDU is received, the transport entity must regard the connection as released.

The transport entity must respond to an invalid CR TPDU by sending an ER or DR TPDU. If an ER TPDU is received in response to a CR TPDU, the initiating transport entity must regard the connection as released.

NOTE

If a supervisory timer TS1 has been set for this connection, then the entity shall stop the timer on receipt of the DR or ER TPDU.

4.4.7 Normal Release

The release procedure is used by a transport entity in order to terminate a transport connection. The implicit variant is used only in class 0. The explicit variant is used in classes 1, 2, 3 and 4. (The variants are discussed below.)

NOTE

When the implicit variant is used (i.e. in class 0), the lifetime of the transport connection is directly correlated with the lifetime of the network connection. The use of the explicit variant of the release procedure enables the transport connection to be released independently of the underlying network connection.

The procedure makes use of the following network service primitives:

- N-DISCONNECT (implicit variant only)
- N-DATA

The procedure makes use of the following TPDUs and fields:

TPDU	Parameters
DR TPDU	clearing reason; user data; SRC-REF; DST-REF.
DC TPDU.	

In the implicit variant, either transport entity disconnects a transport connection by disconnecting the network connection to which it is assigned. Similarly, when a transport entity is informed that the network connection has been disconnected by the peer transport entity, this should be considered as the release of the transport connection.

In the explicit variant, when the release of a transport connection is to be initiated, the transport entity does the following:

a) If it has previously sent or received a CC TPDU (see Note 1 below), it shall send a DR TPDU. It must ignore all subsequently received TPDUs other than a DR or DC TPDU. On receipt of a DR or DC TPDU, it must consider the transport connection released;

b) In other cases, it must wait for acknowledgement of the outstanding CR TPDU. If it receives a CC TPDU, it must follow the above procedure.

c) A transport entity that receives a DR TPDU must, if it has previously sent a DR TPDU for the same transport connection, consider the transport connection released;

d) A transport entity that receives a DR TPDU must, if it has previously sent a CR TPDU that has not been acknowledged by a CC TPDU, consider the connection refused (see subsection 4.4.6).

e) In other cases, it must send a DC TPDU and consider the transport connection released.

NOTE 1

The requirement that a DR TPDU be sent ensures that the transport entity is aware of the remote reference for the transport connection.

NOTE 2

When the transport connection is considered as released, the local reference is either available for re-use or is frozen (see section 4.4.19).

NOTE 3

After the release of a transport connection, the network connection can be released or retained to enable its re-use for the assignment of other transport connections (see section 4.4.1).

NOTE 4

It is recommended that if a transport entity does not receive acknowledgement of a DR TPDU within time TS2, it should either reset or disconnect the network connection, and freeze the reference (see section 4.4.18).

NOTE 5

When a transport entity is waiting for a CC TPDU before sending a DR TPDU and the network connection is reset or released, it should consider the transport connection released and, in classes other than classes 0 and 2, freeze the reference (see section 4.4.18).

4.4.8 Error Release

This procedure is used only in classes 0 and 2 to release a transport connection on the occurrence of a N-DISCONNECT or N-RESET indication. It makes use of the following service primitives:

- N-DISCONNECT indication;
- N-RESET indication.

When on the network connection to which a transport connection is assigned, an N-DISCONNECT or N-RESET indication occurs, both transport entities must consider that the transport connection is released and so inform the TS-users.

NOTE

In other classes, since error recovery is used, the occurrence of an N-RESET indication or N-DISCONNECT indication results in the invocation of the error recovery function.

4.4.9 Association of TPDUs With Transport Connections

This procedure is used in all classes to interpret a received NSDU as TPDU(s) and, if possible, to associate each such TPDU with a transport connection. It makes use of the following network service primitives:

- N-DATA indication;
- N-EXPEDITED DATA indication.

This procedure makes use of the following TPDUs and parameters:

TPDUs	Parameters
any TPDU except CR TPDU, DT TPDU in classes 0 or 1, and AK TPDU in class 1;	DST-REF.

TPDUs	Parameters
CR, CC, DR and DC TPDUs;	SRC-REF.
DT TPDU in classes 0 or 1, and AK TPDU in class 1.	

If the received NSDU or Expedited NSDU cannot be decoded (does not contain one or more correct TPDUs) or is corrupted (contains TPDU with a wrong checksum), then the transport entity must apply to the appropriate procedure for the following:

a) If the network connection on which the error is detected has a class 0 or class 1 transport connection assigned to it, then treat as a protocol error (see section 4.4.22) for that transport connection;

b) Otherwise, carry out one of the following:

   1) ignore the NSDU (in class 4 only);

   2) issue an N-RESET (or N-DISCONNECT) request for the network connection and apply, for all of the transport connections assigned to this network connection (if any), the procedures defined for handling of network signalled reset (and disconnect).

If the NSDU can be decoded and is not corrupted, the transport entity must apply the appropriate one of the following procedures:

c) If the network connection on which the NSDU was received has a class 0 transport connection assigned to it, then consider the NSDU as forming one TPDU and associate the TPDU with the transport connection (described below).

d) Otherwise, invoke the separation procedures and associate each of the individual TPDUs (described below) in the order in which they appear in the NSDU.

If the received TPDU is a CR TPDU, then, if it is a duplicate as recognized by using the NSAPs of the network connection and the SRC-REF parameter it is associated

with the transport connection created by the original value of the CR TPDU. Otherwise, it is processed as requesting the creation of a new transport connection.

If the received TPDU is a DT TPDU and the network connection has a class 0 or 1 transport connection assigned to it, or an AK TPDU where a class 1 transport connection is assigned, then associate the TPDU with the transport connection.

Otherwise, the DST-REF parameter of the TPDU is used to provisionally identify the transport connection. The following cases are distinguished:

a) If the DST-REF is not allocated to a transport connection, the transport entity must respond on the same network connection with a DR TPDU if the TPDU is a CC TPDU, with a DC TPDU if the TPDU is a DR TPDU and must ignore the TPDU if neither a DR TPDU nor CC TPDU. No association with a transport connection is made.

b) If the DST-REF is allocated to a connection, but the TPDU is received on a network connection to which the connection has not been assigned, then there are three cases:

    1) If the transport connection is of class 4, then the TPDU is considered as performing assignment,

    2) If the transport connection is not assigned to any network connection (waiting for reassignment after failure), then association with that transport connection is made.

    3) Otherwise, the TPDU is considered as having a DST-REF not allocated to a transport connection (case a above).

c) If the TPDU is a DC, then it is associated with the transport connection to which the DST-REF is allocated; unless the SRC-REF is not the expected one, in which case the DC TPDU is ignored.

d) If the TPDU is a DR TPDU, then there are three cases:

1) If the SRC-REF is not as expected, then an appropriate DC TPDU is sent back with no association being made;

2) If a CR TPDU is unacknowledged, then the DR TPDU is associated with the transport connection, regardless of the value of its SRC-REF parameter;

3) Otherwise, the DR is associated with the transport connection identified by the DST-REF parameter.

e) If the TPDU is a CC TPDU whose DST-REF parameter identifies an open connection (one for which a CC TPDU has been previously received), and the SRC-REF in the CC TPDU does not match the remote reference, then a DR TPDU is sent back with no association being made.

f) If none of the above cases apply, then the TPDU is associated with the transport connection identified by the DST-REF parameter.

4.4.10 Data TPDU Numbering

Data TPDU numbering is used in classes 1, 2 (when the explicit flow control option is selected), 3 and 4. Its purpose is to enable the use of recovery, flow control and re-sequencing functions. The procedure makes use of the DT TPDU and the parameter TPDU-NR.

A transport entity must allocate the sequence number zero to the TPDU-NR of the first DT TPDU which it transmits for a transport connection. For subsequent DT TPDUs sent for the same transport connection, the transport entity must allocate a sequence number one greater than the previous one.

When a DT TPDU is retransmitted, the TPDU-NR parameter will have the same value as in the first transmission of that DT TPDU.

Modulo 2**7 arithmetic must be used when normal formats have been selected and modulo 2**31 arithmetic must be used when extended formats have been selected. In this chapter, the relationships 'greater than' and 'less than' apply to a set of

contiguous TPDU numbers whose range is less than the modulus and whose startin and finishing numbers are known. The term 'less than' means 'occurring sooner i the window sequence' and the term 'greater than' means 'occurring later in th window sequence'.

4.4.11 Expedited Data Transfer

Expedited data transfer procedures are selected during connection establishmen The network normal data variant may be used in classes 1, 2, 3 and 4. The networ expedited variant is only used in class 1. The procedure makes use of the followin network service primitives:

- N-DATA;
- N-EXPEDITED DATA.

The procedure makes use of the following TPDUs and parameters:

TPDU	Parameters
ED TPDU	ED TPDU-NR;
EA TPDU	YR-TU-NR.

The TS-user data parameter of each T-EXPEDITED DATA request is conveyed a the data field of an Expedited Data (ED) TPDU. Each ED TPDU received must b acknowledged by an Expedited Acknowledge (EA) TPDU. There may only be on ED TPDU unacknowledged at any time for each direction of a transport connection.

An ED TPDU with a zero length data field is a protocol error.

NOTE

The network normal data variant is used, except when the network-expedited variant (available in Class 1 only) has been agreed upon, in which case, ED and EA TPDUs are conveyed in the data fields of N-EXPEDITED DATA primitives. Also, no TPDUs can be transmitted using network-expedited until the

CC TPDU is acknowledged, to prevent the network-expedited from overtaking the CC TPDU.

4.4.12 Reassignment After Failure

The reassignment after failure procedure is used in classes 1 and 3 to commence recovery from an NS-provider signalled disconnect. The procedure uses the N-DISCONNECT indication network service primitive.

When an N-DISCONNECT indication is received for the network connection to which a transport connection is assigned, the initiator must either:

a) Start its TTR timer (see Note 1 below) and assign the transport connection to a different network connection (see section 4.4.1). Until the timer runs out, the transport entity must try to make an assignment for every received N-DISCONNECT indication. If the timer runs out, the transport entity must not try a new assignment. When a valid TPDU is received for the transport connection, the TTR timer is stopped. If no assignment is successful within this time, then the transport connection must be considered released. The reference must be frozen (see section 4.4.18); or

b) Consider the transport connection as released and freeze the reference (see section 4.4.18). (This alternative is only available if a DR TPDU is retained.)

The responder must wait for reassignment to take place for a time TWR following the N-DISCONNECT indication. The arrival of the first TPDU related to the transport connection (and commencing resynchronization; see section 4.4.14) completes the reassignment after failure procedure. If reassignment does not take place within this time, the transport connection is considered released.

If assignment occurs successfully, both transport entities must continue with resynchronization.

NOTE 1

TTR is the Time to Try Reassignment timer. Its value is a local matter, but it must be less than TWR (see Note 2) by at least the sum of the maximum disconnect propagation delay and transit delay of the network connections. Provided that the required quality of service is met, it may be possible to set TTR to zero (i.e. no reassignment); for example, if the rate of NS-provider generated disconnects is very low. The value selected for TTR may be sent to the responder using the reassignment time parameter of the CR TPDU, thereby allowing the responder to use a lower value of TWR. The maximum value of TTR is determined by the default value for TWR (see Note 2).

NOTE 2

TWR is the Time to Wait for Reassignment timer. If the Reassignment Timer parameter is present in the CR TPDU, TWR may be set to this value plus the sum of the maximum disconnect propagation delay and transit delay of the network connections. If the Reassignment Time parameter is not present, a conventional value of two minutes should be used.

4.4.13 Retention Until Acknowledgement of TPDUs

The retention until acknowledgement of TPDUs procedure is used in classes 1, 3 and 4 to enable and minimize retransmission after possible loss of TPDUs.

The confirmation of receipt variant is used only in class 1 when it has been agreed upon during connection establishment (see note below).

The AK variant is used in classes 3 and 4 and also in class 1 when the confirmation of receipt variant has not been agreed upon during connection establishment.

NOTE

Use of confirmation of receipt variant depends on the availability of the network layer receipt confirmation service and the expected cost reduction.

The procedure uses the following network service primitives:

- N-DATA
- N-DATA ACKNOWLEDGE.

It uses the following TPDUs and parameters:

TPDU	Parameters
CR, CC, DR and DC TPDUs	
RJ, AK and EA TPDUs	YR-TU-NR.
DT TPDU	TPDU-NR; ED-TPDU-NR.

Copies of the following TPDUs must be retained upon transmission to permit their later retransmission:

CR, CC, DR, DT and ED TPDUs

except that if a DR is sent in response to a CR TPDU, there is no need to retain a copy of the DR TPDU.

In the confirmation-of-receipt variant applicable only in class 1, transport entities receiving N-DATA indications which convey DT TPDUs and have the confirmation request field set must issue an N-DATA Acknowledge Request (see Notes 1 and 2 below).

After each TPDU is acknowledged, as shown in Table 4-5, the copy need not be retained. Copies may also be discarded when the transport connection is released.

NOTE 1

It is a local matter for each transport entity to decide which N-DATA requests should have the confirmation request parameter set. This decision will normally be related to the amount of storage available for retained copies of the DT TPDUs.

NOTE 2

Use of the confirmation request parameter may affect the quality of network service.

Table 4-5. Acknowledgement of TPDUs

RETAINED TPDU	VARIANT	RETAINED UNTIL ACKNOWLEDGED BY
CR	both	CC, or ER TPDU.
DR	both	DC or DR (in case of collision) TPDU.
CC	confirmation of receipt variant	N-DATA ACKNOWLEDGE INDICATION, RJ, DT, or ED TPDU.
CC	AK variant	RJ, DT, AK, ED or EA TPDU
DT	confirmation of receipt variant	N-DATA ACKNOWLEDGE indication corresponding to an N-DATA request which conveyed, or came after, the DT TPDU.
DT	AK variant	AK or RJ TPDU for which YR-TU-NR is greater than TPDU-NR in the DT TPDU.
ED	both	EA TPDU for which the YR-TU-NR is equal to ED-TPDU-NR in the ED TPDU.

4.4.14 Resynchronization

The resynchronization procedures are used in classes 1 and 3 to restore the transport connection to normal after a reset or disconnect signalled by the NS-provider. The procedure makes use of the N-RESET indication network service primitive.

It uses the following TPDUs and parameters:

TPDU	Parameters
CR, CC, DR and DC TPDUs	
RJ and EA TPDUs	YR-TU-NR
DT TPDU	TPDU-NR
ED TPDU	ED-TPDU-NR

A transport entity which is signalled by the occurrence of an NS-provider generated reset or disconnect must carry out the active resynchronization procedures (described below) unless any of the following hold:

a) The transport entity is the responder and the error was a signalled disconnect (in this case, it is the initiator's responsibility to reassign the connection (see section 4.4.12));

b) The transport entity has an unacknowledged (retained) CC TPDU or has not yet sent the CC because the user has not given the T-CONNECT response.

c) The transport entity has elected not to reassign (see section 4.4.12).

If either (a) or (b) (or both) hold, the transport entity carries out the passive resynchronization procedures. If (c) holds, no resynchronization takes place.

In active resynchronization procedures, the first applicable one of the following actions must be taken:

a) If a CR TPDU is unacknowledged, then the transport entity must retransmit it.

b) If a DR TPDU is unacknowledged, then the transport entity must retransmit it.

c) Otherwise, the transport entity must carry out the data resynchronization procedures described below.

In passive resynchronization procedures, the transport entity must not send any TPDUs until a TPDU has been received. When one has been received, the transport entity must carry out the appropriate one of the following actions, depending on the TPDU:

a) If it is a DR TPDU, then the transport entity must send a DC TPDU;

b) If it is a repeated CR TPDU (see Note 2 below), then the transport entity shall carry out the action which is appropriate among the following:

1) If a CC TPDU has already been sent, and acknowledged, it must be treated as a protocol error;

2) If a DR TPDU is unacknowledged (whether or not a CC TPDU is unacknowledged), retransmit the DR TPDU with the source reference set to zero;

3) If the T-CONNECT response has not yet been received from the user, take no action;

4) Otherwise, retransmit the CC TPDU followed by any unacknowledged ED TPDU (see Note 3 below) and any DT TPDU;

c) If it is an RJ TPDU, then the first applicable one of the following actions must be taken:

1) If a DR TPDU is unacknowledged, then the transport entity must retransmit it;

2) Otherwise, the transport entity must carry out the data resynchronization procedures (described below).

NOTE 1

If a CC TPDU was unacknowledged, the RJ TPDU should then be considered as acknowledging the CC TPDU. If a CC TPDU was never sent, the RJ TPDU should then be considered as a protocol error.

NOTE 2

A repeated CR can be identified by being on a network connection with the appropriate network addresses and having a correct source reference.

NOTE 3

The transport entity should not use network-expedited until the CC is acknowledged (see section 4.4.5). This rule prevents the network expedited from overtaking the CC TPDU.

NOTE 4

The RJ TPDU may have reduced the credit.

In data resynchronization, the transport entity must carry out the following actions in the order listed:

a) Transmit RJ TPDU with YR-TU-NR field set to the TPDU-NR of the next expected DT TPDU;

b) Wait for the RJ TPDU from the other transport entity, unless it has already been received; if a DR TPDU is received, the transport entity must send a DC, inform the TS-user of the disconnection and take no further action (i.e. it must not follow the procedures in c, d and e above);

c) (Re)transmit any ED TPDU which is unacknowledged;

d) (Re)transmit any DT TPDUs which are unacknowledged, subject to any applicable flow control procedures (see note 4 above);

e) If any duplicate ED TPDUs are received, the transport entity shall acknowledge them with an EA TPDU and then discard the duplicated ED TPDU.

4.4.15 Multiplexing and Demultiplexing

The multiplexing and demultiplexing procedures are used in classes 2, 3 and 4 to allow several transport connections to share a network connection at the same time. The procedure makes use of the following TPDUs and parameters:

TPDU	Parameters
CC, DR, DC, DT, AK, ED, EA, RJ and ER TPDUs	DST-REF

The transport entities must be able to send and receive, on the same network connection, TPDUs belonging to different transport connections.

NOTE 1

When performing demultiplexing, the transport connection to which the TPDUs apply must be distinguished by the DST-REF parameter except in the case of a CR TPDU (see section 4.4.9).

NOTE 2

Multiplexing allows the concatenation of TPDUs belonging to different transport connections that are to be transferred in the different transport connections to be transferred in the same N-DATA primitive (see section 4.4.4).

4.4.16 Explicit Flow Control

The explicit flow control procedure is used in classes 2, 3 and 4 to regulate the flow of DT TPDUs independently of the flow control in the other layers. The procedure makes use of the following TPDUs and parameters:

TPDU	Parameters
CR, CC, AK and RJ TPDUs	CDT
DT TPDU	TPDU-NR
AK and RJ TPDUs	YR-TU-NR subsequent number flow control confirmation

The procedures differ in different classes. They are defined in the following sections (4.5 - 4.9) specifying the separate classes.

4.4.17 Checksum

The checksum procedure is used to detect corruption of TPDUs by the NS-provider. The checksum is used in class 4 only for the CR TPDU and for others except if non-use of the procedure was agreed during connection establishment.

The procedure uses all TPDUs and parameter checksum.

The sending transport entity must transmit TPDUs with the checksum parameters set such that the following formulas are satisfied:

$$\sum_{i=1}^{L} a_i \equiv 0 \pmod{255}$$

$$\sum_{i=1}^{L} i a_i \equiv 0 \pmod{255}$$

where

i = number (i.e. position) of an octet within the TPDU (see section 4.10);

a_i = value of octet in position i;

L = length of TPDU in octets.

A transport entity which receives a TPDU for a transport connection for which the checksum has been agreed upon and which does not satisfy the above formulas must discard the TPDU (see also Note 2 below).

NOTE 1

An efficient algorithm for determining the checksum parameters is given in section 4.14.

NOTE 2

Since it is not possible, in this case, to know with certainty the transport connection to which the TPDU relates, further action may be taken for all the transport connections assigned to the network connection (see section 4.4.9).

4.4.18 Frozen References

This procedure is used in order to prevent re-use of a reference while TPDUs associated with the old use of the reference may still exist.

When a transport entity determines that a particular connection is released, it must place the reference which is allocated to the connection in a frozen state. While frozen, the reference must not be re-used. The period of time for which the reference remains frozen depends on the class (see Note 1 below).

Where the class of the transport connection includes resynchronization (i.e. classes 1 and 3), there are only three cases where the release of the connection does not require the freezing of a reference. These are:

a) where the transport entity receives a DC TPDU in response to a DR TPDU which it has sent (see Note 2 below);

b) where the transport entity sends a DR TPDU in response to a CR TPDU which it has received (see Note 3 below);

c) where the transport entity has considered the connection to be released after the expiration of the TWR timer (see Note 4 below).

NOTE 1

This function is necessary because retransmission or misordering can cause TPDUs bearing a reference to arrive at an entity after it has released the connection for which it allocated the reference. Retransmission, for example, can arise when the class includes either resynchronization (see section 4.4.14) or retransmission on time out (see section 4.4.19). Freezing of the reference is never necessary in classes 0 or 2 (although it can be done as a local decision). It is always mandatory in class 4.

NOTE 2

In this case, it is certain that the other transport entity considers the connection released.

NOTE 3

In this case, the other transport entity has not been informed of any reference assignment and, thus, cannot possibly make use of a

reference (this includes the case where a CC TPDU was sent, but was lost).

NOTE 4

In this case, the transport entity has already effectively frozen the reference for an adequate period.

4.4.19 Retransmission on Time-Out

The procedure is used in class 4 to cope with unsignalled loss of TPDUs by the NS-provider. The procedure makes use of the following TPDUs:

CR, CC, DR, DT, ED, AK.

The procedure is specified in the procedures for class 4.

4.4.20 Resequencing

The resequencing procedure is used in class 4 to cope with misordering of TPDUs by the network service provider. The procedure uses the following TPDUs and fields:

TPDU	Parameters
DT TPDU	TPDU-NR
ED TPDU	ED TPDU-NR

The procedure is specified in the procedures for class 4.

4.4.21 Inactivity Control

The inactivity control procedure is used in class 4 to cope with unsignalled termination of a network connection. The procedure is specified in the procedures for class 4 (section 4.10.3).

4.4.22 Treatment of Protocol Errors

The procedure for treatment of protocol errors is used in all classes to deal with invalid TPDUs. The procedure uses the following TPDUs and parameters:

TPDU	Parameters
ER TPDU	Reject cause TPDU in error
DR TPDU	Reason code

A transport entity that receives a TPDU that can be associated with a transport connection and is invalid or constitutes a protocol error (e.g. containing an invalid parameter or value) must take appropriate action so it won't jeopardize any other transport connections not assigned to that network connection. The appropriate actions must include one of the following actions:

a) Ignoring the TPDU;

b) Transmitting an ER TPDU;

c) Resetting or closing the network connection; or

d) Invoking the release procedures appropriate to the class.

If an ER TPDU is set in class 0, it must contain the octets of the invalid TPDU up to and including the octet where the error was detected (see Notes 3, 4 and 6 below).

If the TPDU cannot be associated to a particular transport connection, then see section 4.4.9.

NOTE 1

In general, no further action is specified for the receiver of the ER TPDU, but it is recommended that it initiates the release procedure appropriate to the class. If the ER TPDU has been received as an answer to a CR TPDU, then the connection is regarded as released (see section 4.4.6).

NOTE 2

Care should be taken by a transport entity receiving several invalid TPDUs or ER TPDUs to avoid looping if the error is generated repeatedly.

NOTE 3

If the invalid received TPDU is greater than the selected maximum TPDU size, it is possible that it cannot be included in the TPDU-in-error parameter of the ER TPDU.

NOTE 4

It is recommended that the sender of the ER TPDU start a timer TS2 to ensure the release of the connection. If the timer expires, the transport entity must initiate the release procedures appropriate to the class.

NOTE 5

A TPDU which is defined in this standard but is not to be used when a particular class is in operation can either be regarded as an unknown TPDU or known TPDU violating the procedures of the class.

NOTE 6

In classes other than 0, it is recommended that the invalid TPDU be also included in the ER TPDU.

4.4.23 Splitting and Recombining

This procedure is used in class 4 only to allow a transport connection to make use of multiple network connections to provide additional resilience against network failure, to increase throughput, or for other reasons.

When this procedure is being used, a transport connection can be assigned (see section 4.4.1) to multiple network connections (see Note 1 below). TPDUs for the connection may be sent over any such network connection.

If the use of class 4 is not accepted by the remote transport entity following th negotiation rules, then no network connection, except that over which the CR TPD was sent, may have this transport connection assigned to it.

NOTE 1

The resequencing function of class 4 (see section 4.4.20) is used to ensure that TPDUs are processed in the correct sequence.

NOTE 2

Either transport may assign the connection to further network connections of which it is the owner at any time during the life of the transport connection.

NOTE 3

In order to enable the detection of unsignalled NC failures, a transport entity performing splitting should ensure that TPDUs are sent at intervals on each supporting network connection, for example, by sending successive TPDUs on each network connection. By monitoring each network connection, a transport entity may detect unsignalled network connection failures, following the inactivity procedures defined in section 4.10.3. Thus, for each network connection, no period (see section 4.10.3) may elapse without the receipt of some TPDU for some transport connection.

4.5 PROTOCOL CLASSES

Table 4-6 gives an overview of which elements of procedure are included in eac class. In certain cases, the elements of procedure within different classes are n identical and, for this reason, Table 4-6 cannot be considered as part of th definitive specification of the protocol.

The following symbols are used in Table 4-6:

Symbol	Meaning
*	Procedure always included in class.
	Not applicable
m	Negotiable procedure whose implementation in equipment is mandatory.
o	Negotiable procedure whose implementation in equipment is optional.
ao	Negotiable procedure whose implementation in equipment is optional and where use depends on availability within the network service.
(1)	Not applicable in class 2 when non-use of explicit flow control is selected.

Table 4-6. Procedures Included in Each Class

Protocol Mechanism	Cross Reference	Variant	0	1	2	3	4
Assignment to network Conn	4.4.1		*	*	*	*	*
TPDU Transfer	4.4.2		*	*	*	*	*
Segmenting and Reeassembling	4.4.3		*	*	*	*	*
	4.4.4			*	*	*	*
Connection Establishment	4.4.5		*	*	*	*	*
Connection Refusal	4.4.6		*	*	*	*	*
Normal Release	4.4.7	implicit explicit	*	*	*	*	*
Error Release	4.4.8		*		*		
Association of TPDUs with Transport Connection	4.4.9		*	*	*	*	*
DT TPDU Numbering	4.4.10	Normal extended		*	m(1) 0(1)	m 0	m 0
Expedited Data Transfer	4.4.11	network normal network expedited		m ab	* (1)	*	*
Reassignment after failure	4.4.12			*		*	

Table 4-6. Procedures Included in Each Class (Continued)

Protocol Mechanism	Cross Reference	Variant	0	1	2	3	4
Retention until Acknowledgement of TPDUs	4.4.13	conf.receipt		ao			
		AK		m		*	*
Resynchronization	4.4.14			*		*	
Multiplexing and Demultiplexing	4.4.15				*	*	*
Explicit Flow control with	4.4.16				m	*	*
without			*	*	0		
Checksum (use of)	4.4.17						m
(non-use of)			*	*	*	*	o
Frozen References	4.4.18			*		*	*
Retransmission on Timeout	4.4.19						*
Resequencing	4.4.20						*
Inactivity Control	4.4.21						*
Treatment of Protocol Errors	4.4.22		*	*	*	*	*
Splitting and Recombining	4.4.23						*

4.6 SPECIFICATION FOR CLASS 0. (SIMPLE CLASS)

Class 0 is designed to have minimum functionality. It provides only the functions needed for connection establishment with negotiation, data transfer with segmenting, and protocol error reporting. This class provides transport connections with flow control based on the network service provided flow control, and disconnection based on the network service disconnection.

4.6.1 Procedures Applicable at All Times

The transport entities must use the following procedures (applicable at all times):

a) TPDU transfer (see section 4.4.2);

b) Association of TPDUs with transport connections (see section 4.4.9);

c) Treatment of protocol errors (see section 4.4.22);

d) Error release (see section 4.4.8).

4.6.2 Connection Establishment

To establish connection, the transport entities must use the following procedures:

a) Assignment to network connection (see section 4.4.1); and then

b) Connection establishment (see section 4.4.5) and, if appropriate, connection refusal (see section 4.4.6);

Subject to the following constraints:

- The CR and CC TPDUs must contain no parameter field other than those for TSAP-ID and maximum TPDU size;
- The CR and CC TPDUs must not contain a data field.

4.6.3 Data Transfer

For data transfer, the transport entities must use the segmenting and reassembling procedure (see section 4.4.3).

4.6.4 Release

For release, the transport entities must use the implicit variant of the normal release procedure (see 4.4.7).

NOTE

The lifetime of the transport connection is directly correlated with the lifetime of the network connection.

4.7 SPECIFICATION FOR CLASS 1 (BASIC ERROR RECOVERY CLASS)

Class 1 provides transport connections with flow control based on network service provided flow control, error recovery, expedited data transfer, disconnection, and also the ability to support consecutive transport connections on a network connection.

This class provides the functionality of class 0 plus the ability to recover after a failure signalled by the Network Service, without involving the TS-user.

4.7.1 Procedures Applicable at All Times

The transport entities must use the following procedures:

a) TPDU transfer (see section 4.4.2);

b) Association of TPDU with transport connections (see section 4.4.9);

c) Treatment of protocol errors (see section 4.4.22);

d) Reassignment after failure (see section 4.4.12);

e) Resynchronization (see section 4.4.14), or reassignment after failure (see section 4.4.12) together with resynchronization (see section 4.4.14);

f) Concatenation and separation (see section 4.4.4);

g) Retention until acknowledgement of TPDU (see section 4.4.13); The variant used, AK or confirmation of receipt, must be as selected during connection establishment (see Notes below);

h) Frozen references (see section 4.4.18).

NOTE 1

The negotiation of the variant of retention until acknowledgement of TPDUs procedure to be used over the transport connection is designed such that if the initiator proposes the use of the AK variant (i.e. the mandatory implementation option), the responder has to accept use of this option; and, if the initiator proposes use of the confirmation of receipt variant, the responder is entitled to select use of the AK variant.

NOTE 2

The AK variant makes use of AK TPDUs to release copies of retained DT TPDUs. The CDT parameter of AK TPDUs in class 1 is not significant, and is set to 1111.

NOTE 3

The confirmation of receipt variant is restricted to this class and its use depends on the availability of the network layer receipt confirmation service, and the expected cost reduction.

4.7.2 Connection Establishment

The transport entities must use the following procedures:

a) Assignment to network connection (see section 4.4.1); then

b) Connection establishment (see section 4.4.5) together with connection refusal (see section 4.4.6).

4.7.3 Data Transfer

The sending transport entity must use the following procedures:

a) Segmenting (see section 4.4.3); and then

b) The normal format variant of DT TPDU numbering (see section 4.4.10).

The receiving transport entity must use the following procedures:

c) The normal variant of DT TPDU numbering (see section 4.4.10); then

d) Reassembling (see section 4.4.3).

NOTE 1

The use of RJ TPDU during resynchronization (see section 4.4.14) can lead to retransmission. Thus, the receipt of a DT TPDU, which is a duplicate, is possible and is, therefore, discarded.

NOTE 2

It is possible to decide on a local basis to issue an N-RESET request in order to force the remote entity to carry out the resynchronization (see section 4.4.14).

For expedited data, the transport entities must use either the network normal data or the network expedited variants of the expedited data transfer procedure (see section 4.4.11) if their use has been selected during connection establishment (see Note 1 below).

The sending transport entity must not allocate the same ED TPDU-NR to successive ED TPDUs (see Notes 2 and 3 below).

NOTE 1

The negotiation of the variant of expedited data transfer procedure to be used over the transport connection is designed such that if the initiator proposes the use of the network normal data variant (i.e. the mandatory implementation option), the responder has to accept use of this option; and, if the initiator proposes use of the network expedited variant, the responder is entitled to select use of the network normal data variant.

NOTE 2

This numbering enables the receiving transport entity to discard repeated ED TPDUs when resynchronization (see section 4.4.14) has taken place.

NOTE 3

No other significance is attached to the ED TPDU-NR parameter. It is recommended, but not essential, that the values used be consecutive modulo 128.

4.7.4 Release

The transport entities must use the explicit variant of the release procedure (see section 4.4.7).

4.8 SPECIFICATION FOR CLASS 2 (MULTIPLEXING CLASS)

Class 2 provides transport connections with or without individual flow control - no error detection or error recovery is provided.

If the network resets or clears, the transport connection is terminated without the transport release procedure and the TS-user is informed.

When explicit flow control is used, a credit mechanism is defined allowing the receiver to inform the sender of the exact amount of data the receiver is willing to receive, and expedited data transfer is available.

4.8.1 Procedures Applicable at All Times

The transport entities must use the following procedures:

a) Association of TPDUs with transport connection (see section 4.4.9);

b) TPDU transfer (see section 4.4.2);

c) Treatment of protocol errors (see section 4.4.22);

d) Concatenation and separation (see section 4.4.4);

e) Error release (see section 4.4.8).

Additionally, the transport entities may use the multiplexing procedure (see section 4.4.15).

4.8.2 Connection Establishment

The transfer entities must use the following procedures:

a) Assignment to network connection (see section 4.4.1); then

b) Connection establishment (see section 4.4.5) and, if applicable, connection refusal (see section 4.4.6).

4.8.3 Data Transfer When Non-Use of Explicit Flow Control Has Been Selected

If this option is selected as a result of the connection establishment, the transport entities must use the segmenting procedure (see section 4.4.3).

The TPDU-NR field of DT TPDUs is not significant and may take any value.

NOTE

Expedited data transfer is not applicable (see section 4.4.5).

4.8.4 Data Transfer When Use of Explicit Flow Control Has Been Selected

The sending transport entity must use:

a) segmenting (see section 4.4.3); then

b) DT TPDU numbering (see section 4.4.10);

The receiving transfer entity must use:

a) DT TPDU numbering (see section 4.4.10) (if a DT TPDU is received which is out of sequence, it shall be treated as a protocol error) and then

b) reassembling (see section 4.4.3).

The variant of the DT TPDU numbering which is used by both transport entities must be that which was agreed upon at connection establishment.

For flow control, the transport entities must send an initial credit (which may be 0) in the CDT field of the CR or CC TPDU. This credit represents the initial value of the upper window edge allocated to the peer entity.

The transport entity that receives the CR or the CC TPDU must consider its lower window edge as zero, and its upper window edge as the value of the CDT field in the received TPDU.

In order to authorize the transmission of DT TPDUs, by its peer, a transport entity may transmit an AK TPDU at any time, subject to the following constraints:

a) The YR-TU-NR field must be at most one greater than the TPDU-NR field of the last received DT TPDU or must be zero if no DT TPDU has been received;

b) The YR-TU-NR field must not be lower than that in the previously sent AK TPDU, or lower than zero if no AK TPDUs have been sent;

c) The sum of the YR-TU-NR and CDT fields must not be lower than the upper window edge allocated to the remote entity (see Note 1 below).

A transport entity which receives an AK TPDU must consider the YR-TU-NR field as its new lower window edge, and the sum of YR-TU-NR and CDT as its new upper window edge. If either of these have been reduced or if the lower window edge has become more than one greater than the TPDU-NR of the last transmitted DT TPDU, this must be treated as a protocol error (see section 4.4.22).

A transport entity must not send a DT TPDU with a TPDU-NR outside of the transmit window (see Notes 2 and 3 below).

NOTE 1

This means that credit reduction is not applicable.

NOTE 2

This means that a transport entity is required to stop sending if the TPDU-NR field of the next DT TPDU which would be sent would be the upper window edge. Sending of DT TPDU may be resumed if an AK TPDU is received which increases the upper window edge.

NOTE 3

The rate at which a transport entity progresses the upper window edge allocated to its peer entity constrains the throughput attainable on the transport connection.

For expedited data, the transport entities must follow the network normal variant of the expedited data transfer procedure in section 4.4.1 if its use has been agreed upon during connection establishment. ED and EA TPDUs are not subject to the flow control procedures described above. The ED-TPDU-NR and YR-TU-NR fields of ED and EA TPDUs, respectively, are not significant and may take any value.

4.8.5 Release

The transport entities must use the explicit variant of the release procedure in section 4.4.7.

4.9 SPECIFICATION FOR CLASS 3 (ERROR RECOVERY AND MULTIPLEXING CLASS)

Class 3 provides the functionality of class 2 (with use of explicit flow control) plus the ability to recover after a failure signalled by the Network Layer without involving the user of the transport service. The mechanisms used to achieve this functionality also allow the implementation of more flexible flow control.

4.9.1 Procedures Applicable at All Times

The transport entities must use the following procedures:

a) Association of TPDUs with transport connections (see section 4.4.9);

b) TPDU transfer (see section 4.4.2) and retention until acknowledgement of TPDUs (AK variant only) (see section 4.4.13);

c) Treatment of protocol errors (see section 4.4.22);

d) Concatenation and separation (see section 4.4.4);

e) Reassignment after failure (see section 4.4.12), together with resynchronization (see section 4.4.14);

f) Frozen references (see section 4.4.18).

Additionally, the transport entities may use the multiplexing procedure (see section 4.4.15).

4.9.2 Connection Establishment

The transport entities must use the following procedures:

a) Assignment to network connections (see section 4.4.1); and then

b) Connection establishment (see section 4.4.5) together with connection refusal (see section 4.4.6).

4.9.3 Data Transfer

The sending transport entity must use the following procedures:

a) Segmenting (see section 4.4.3); then

b) DT TPDU numbering (see section 4.4.10); after receipt of an RJ TPDU (discussed below) the next DT TPDU to be sent may have a value which is not the previous value of TPDU-NR plus one.

The receiving transport entity must use the following procedures:

c) DT TPDU numbering (see section 4.4.10); the TPDU-NR field of each received DT TPDU must be treated as a protocol error if it exceeds the greatest such value received in a previous DT TPDU by more than one (see Note below); and then

d) Reassembling (see section 4.4.3). Duplicated TPDUs must be eliminated before reassembling is performed.

NOTE

The use of RJ TPDUs (discussed below) can lead to retransmission and reduction of credit. Thus, receipt of a DT TPDU which is a duplicate or which is greater than the upper window edge allocated to the peer entity is possible, and is, therefore, not treated as a protocol error.

A transport entity may send an RJ TPDU at any time in order to invite retransmission or to reduce the upper window edge allocated to the peer entity (see Note 1 below).

When an RJ TPDU is sent, the following constraints must be respected:

a) The YR-TU-NR parameter must be at most one greater than the greatest such value received in a previous DT TPDU, or be zero if no DT TPDU has yet been received (see Note 2 below);

b) The YR-TU-NR parameter must not be lower than that in the previously sent AK or RJ TPDU or lower than zero if no AK or RJ TPDUs has been sent.

When a transport entity receives such an RJ TPDU (see Note 3 below):

a) The next DT TPDU to be transmitted, or retransmitted, must be that for which the value of the TPDU-NR parameter is equal to the value of the YR-TU-NR parameter of the RJ TPDU;

b) The sum of the values of the YR-TU-NR and CDT-parameters of the RJ TPDU becomes the new upper window edge (see Note 4 below).

NOTE 1

An RJ TPDU can also be sent as part of the resynchronization (see section 4.4.14) and reassignment after failure (see section 4.4.12) procedures.

NOTE 2

It is recommended that the YR-TU-NR parameter be equal to the TPDU-NR parameter of the next expected DT TPDU.

NOTE 3

These rules are a subset of those specified for when an RJ TPDU is received during resynchronization (see section 4.4.14) and reassignment after failure (see section 4.4.12).

NOTE 4

This means that RJ TPDU can be used to reduce the upper window edge allocated to the peer entity (credit reduction).

For flow control, the procedures must be as defined in section 4.8.3, except that:

a) Receipt of a DT TPDU with a TPDU-NR parameter whose value is not, but would have been but for a credit reduction, less than the upper window edge allocated to the remote entity, must not be treated as a protocol error;

b) Receipt of an AK TPDU which sets the lower window edge more than one greater than the TPDU-NR of the last transmitted DT TPDU must not be treated as a protocol error, provided that all acknowledged DT TPDUs have been previously transmitted (see Notes 5 and 6 below).

NOTE 5

This can only occur during retransmission following receipt of an RJ TPDU.

NOTE 6

The transport entity may either continue retransmission as before, or retransmit only those DT TPDUs not acknowledged by the AK TPDU. In either case, copies of the acknowledged DT TPDUs need not be retained further.

For expedited data, the transport entities must follow the network normal data variant of expedited data transfer procedure in section 4.4.11 if its use has been agreed upon during connection establishment.

The sending transport entity must not allocate the same ED-TPDU-NR to successive ED TPDUs.

The receiving transport entity must transmit an EA TPDU with the same sequence number in its ED-TPDU-NR field. If, and only if, this number is different from that of the previously received ED TPDU, it must generate a T-EXPEDITED DATA indication to convey the data to the TS-user (see Note 8 below).

NOTE 7

No other significance is attached to the ED-TPDU-NR field. It is recommended, but not essential, that the values be consecutive modulo 2^n, where n is the number of bits in the sequence number field.

NOTE 8

This procedure ensures that the TS-user does not receive data corresponding to the same ED TPDU more than once.

4.9.4 Release

The transport entities must use the explicit variant of the release in section 4.4.7.

4.10 SPECIFICATION FOR CLASS 4 (ERROR DETECTION AND RECOVERY CLASS)

Class 4 provides the functionality of class 3, plus the ability to detect and recover from lost, duplicated, or out of sequence TPDUs without involving the TS-user. This detection of errors is made by extended use of the DT TPDU numbering of class 2 and class 3, by time-out mechanisms, and by additional procedures.

This class additionally detects and recovers from damaged TPDUs by using a checksum mechanism. The use of the checksum mechanism must be available, but its use or its non-use is subject to negotiation. Class 4 does not attempt to deal with detection of errors due to the misdelivery of TPDUs.

The R1 version of iNA 960 falls into this class.

4.10.1 Procedures Applicable at All Times

Many procedures in this class use timers. The following define these timers (see also Table 4-7):

Timers that apply only to specific procedures are defined under the appropriate procedure.

NSDU lifetime (M): The network layer is assumed to provide, as an aspect of its grade of service, a bound on the maximum lifetime of NSDUs in the network. This value is assumed to be known by the transport entities and is the maximum time which may elapse between the transmission of an NSDU to the network layer and receipt of any copy of it.

Expected maximum transit delay (E): It is assumed that the transport entities know the value for the expected maximum transit delay of the network, which will be the maximum delay suffered by all but a small proportion of NSDUs.

Table 4-7. Timer Parameters Related to the Operation of Class 4

Symbol	Name	Definition
M	NSDU lifetime	A bound for the maximum time which may elapse between the transmission of an NSDU by a transport entity and the receipt of any copy of it by its peer entity.
E	Expected maximum transmit delay	A bound for the maximum delay suffered by all but a small proportion of NSDUs.
A_L	Local acknowledge time	A bound for the maximum time which can elapse between the receipt of a TPDU by the local transport entity from the network layer and the transmission of the corresponding acknowledgement.
A_R	Remote acknowledgement time	Same as A_L, but for the remote entity.
T1	Local retransmission time	A bound for the maximum time the local transport entity will wait for acknowledgement before re-transmitting a TPDU.
R	Persistence time	A bound for the maximum time that the local transport entity will continue to transmit a TPDU that requires acknowledgement.
N	Maximum number of transmissions	A bound for the maximum number of times which the local transport entity will continue to transmit a TPDU that request acknowledgement.
L	Bound on reference identifier and sequence number	A bound for the maximum time between the transmission of a TPDU and the receipt of any acknowledgment relating to it.
I	Inactivity time	A bound for the time after which a transport entity will, if it does not receive a TPDU, initiate the release procedure to terminate the transport connection. NOTE: This parameter is required for protection against unsignalled breaks in the network connection.
W	Window time	A bound for the maximum time a transport entity will wait before retransmitting up to date window information.

Acknowledge Time: Any transport entity is assumed to provide a bound for the maximum time which can elapse between its receipt of a TPDU from the Network Layer and its transmission of the corresponding response. This value is referred to as A_L. The corresponding time given by the remote transport entity is referred to as A_R.

Local retransmission time (T1): The local transport entity is assumed to maintain a bound on the time it will wait for an acknowledgement before retransmitting the TPDU. Its value is given by:

$$T1 = 2 * E + A_R + x$$

where:

E = Expected maximum transit delay,
A_R = Remote acknowledge time, and
x = Local processing time for a TPDU.

Persistence Time (R): The local transport entity is assumed to provide a bound for the maximum time for which it may continue to retransmit a TPDU requiring positive acknowledgement. This value is referred to as R.

The value is clearly related to the time elapsed between retransmission, T1, and the maximum number of transmissions, N. It is not less than T1 * N + X, where X is a small quantity to allow for additional internal delays, the granularity of the mechanism used to implement T1 and so on. Because R is a bound, the exact value of X is unimportant as long as it is bounded and the value of a bound is known.

Bound on Reference Identifier and Sequence Number (L): A bound for the maximum time between the decision to transmit a TPDU and the receipt of any response relating to it (L) is given by:

$$L = 2 * M + R + A_R$$

It is necessary to wait for a period L before reusing any reference or sequence number, to avoid confusion in case a TPDU referring to it may be duplicated or delayed.

NOTE 1

In practice, the value of L may be unacceptably large. It may also be only a statistical figure at a certain confidence level. A smaller value may, therefore, be used where this still allows the required quality of service to be provided.

NOTE 2

The relationships between times discussed above are illustrated in figures 4-2 and 4-3.

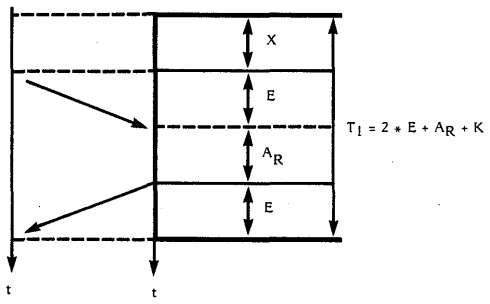


Figure 4-2. The Interrelationship of Times For the Average Case in Class 4

The transport entity must use the following procedures:

a) TPDU transfer (see section 4.4.2);

b) Association of TPDUs with network connections (see section 4.4.10);

c) Treatment of protocol errors (see section 4.4.22);

d) Checksum (see section 4.4.17);

e) Splitting and recombining (see section 4.4.23);

f) Multiplexing and demultiplexing (see section 4.4.15);

g) Retention until acknowledgement (see section 4.4.13);

h) Frozen references (see section 4.4.18).

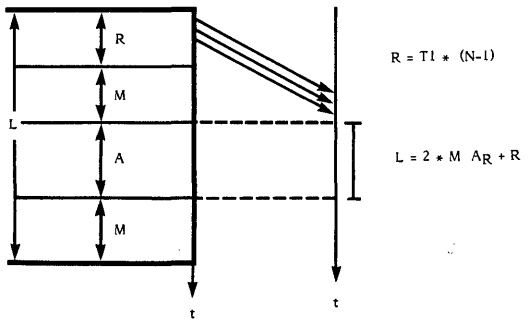


Figure 4-3. The Interrelationship of Times For Maximum Delay in Class 4

The transport entities must use the retransmission procedures as follows:

When a transport entity has some outstanding TPDUs that require acknowledgement, it will check to ensure that no T1 interval elapses without the arrival of a TPDU that acknowledges at least one of the outstanding TPDUs.

If the timer expires, the first TPDU is retransmitted and the timer is restarted. After N transmissions (i.e. N-1 retransmissions), it is assumed that useful two-way communication is no longer possible, the release procedure is used, and the transport user is informed.

This procedure may be implemented by different means. For example:

a) One interval is associated with each TPDU. If the timer expires, the associated TPDU will be transmitted and the timer T1 will be restarted for all subsequent TPDUs; or

b) One interval is associated with each transport connection:

   1) If the transport entity transmits a DT TPDU requiring acknowledgement, it starts timer T1;

   2) If the transport entity receives a TPDU that acknowledges one of the TPDUs to be acknowledged, it restarts timer T1;

   3) If the transport entity receives a TPDU that acknowledges the last TPDU to be acknowledged, it stops timer T1.

For a decision whether the retransmission timer T1 is maintained on a per TPDU or on a per transport connection basis, throughput considerations have to be taken into account.

It is recommended that after N transmissions, the transport entity waits T1 + I to provide the greatest possibility of receiving an acknowledgement before entering the release phase.

4.10.2 Connection Establishment

There are no timers specific to connection establishment. The transport entities must use the following procedures:

a) Assignment to network connection (see section 4.4.1);

b) Connection establishment (see section 4.4.5) together with connection refusal (see section 4.4.6) together with the following additional procedures:

1) A connection is not considered established until the successful completion of a 3-way TPDU exchange. The sender of a CR TPDU must respond to the corresponding CC TPDU by immediately sending a DT, ED or AK TPDU;

2) As a result of duplication or retransmission, a CR TPDU may be received specifying a source reference which is already in use with the sending transport entity. If the receiving transport entity is in the data transfer phase, having completed the 3-way TPDU exchange procedure, the receiving transport entity should ignore such a TPDU. Otherwise, a CC TPDU should be transmitted;

3) As a result of duplication or retransmission, a CC TPDU may be received specifying a paired reference which is already in use. The receiving transport entity should ignore such a CC TPDU;

4) A CC TPDU may be received specifying a reference which is in the frozen state. The response to such a TPDU should be a DR TPDU;

5) The retransmission procedures (see section 4.10.1) are used for both the CR TPDU and CC TPDU.

4.10.3 Data Transfer

The data transfer procedures use two additional timers:

Inactivity Time (I): To protect against unsignalled breaks in the network connection or failure of the peer transport entity (half-open connections), each transport entity maintains an inactivity interval. The interval must be greater than E.

NOTE

A suitable value for I is given by N.R. unless local needs indicate another more appropriate value.

Window Time (W): A transport entity maintains a timer interval to ensure that there is a bound on the maximum interval between window updates.

NOTE

To prevent unintended break of TC, a suitable value for W is given by:

W (remote value of I) /N

The transport entities must use the following procedures:

a) Inactivity control (see section 4.4.21);

b) Expedited data (see section 4.4.11);

c) Explicit flow control (see section 4.4.16).

The sending transport entity must use the following procedures in the order listed:

a) Segmenting (see section 4.4.3);

b) DT TPDU numbering (see section 4.4.10).

The receiving transport entity must use the following procedures in the order listed:

a) DT TPDU numbering (see section 4.4.10);

b) Resequencing (see section 4.4.20);

c) Reassembly (see section 4.4.3).

For inactivity control, if the interval of the inactivity timer I expires without receipt of some TPDU, the transport entity will terminate the transport connection by making use of the release procedures. To prevent expiration of the remote transport entity's inactivity timer when no data is being sent, the local transport entity must send AK TPDUs at suitable intervals in the absence of data, having regard for the probability of TPDU loss. The window synchronization procedures (discussed below) may ensure that this requirement is met.

NOTE

It is likely that the release procedure initiated due to the expiration of the inactivity timer will fail, as such expiration indicates probable failure of the supporting network connection or of the remote transport entity.

For expedited data, the transport entities must follow the network normal variant of the expedited data transfer procedures (see section 4.4.11), if it has been agreed upon during connection establishment.

The ED TPDU must have a TPDU-NR which is allocated from a sequence space separate from that of the DT TPDUs. The EA TPDU carries the same value in its YR-TU-NY as the corresponding ED TPDU. Only a single ED TPDU is to be transmitted and awaiting acknowledgements at any time.

The receiving transport entity must transmit an EA TPDU with the same sequence number in its ED-TPDU-NR field. If this number is one greater than in the previously received ED TPDU, the receiving transport entity must transfer the data in the ED TPDU to the TS-user.

If a transport entity does not receive an EA TPDU in acknowledgement to an ED TPDU, it must follow the retransmission procedures (see note in section 4.10.1).

The sender of an ED TPDU must not send any new DT TPDU with higher TPDU-NR until it receives the EA TPDU. This guarantees the arrival of the ED TPDU before any subsequently sent DT TPDUs.

NOTE

This procedure ensures that ED TPDUs are delivered to the TS-user in sequence and that the TS-user does not receive data corresponding to the same ED TPDU more than once.

For resequencing, the receiving transport entity must maintain the proper sequence of DT TPDUs. DT TPDUs received out-of-sequence must not be delivered to the TS-user until all in-sequence TPDUs have been received.

The transport entity must perform this procedure by ensuring that all DT TPDUs are delivered to the TS-user in the order specified by the sequence number field.

Duplicate TPDUs can be detected because the sequence number matches that of previously received TPDUs. Sequence numbers must not be reused for the period L after their previous use. Otherwise, a new, valid TPDU could be confused with a duplicated TPDU which was previously received and acknowledged.

Duplicated DT TPDUs must be acknowledged, since the duplicated TPDU may be the result of a retransmission resulting from the loss of an AK TPDU. The data contained in a duplicated DT TPDU should be ignored.

For explicit flow control, the transport entities must send an initial credit (which may take the value 0) in the CDT field of the CR TPDU or CC TPDU. This credit represents the initial value of the upper window edge of the peer entity. The transport entity which receives the CR TPDU or CC TPDU must consider its lower window edge as zero and its upper window edge as the value in the CDT field in the received TPDU.

In order to authorize the transmission of DT TPDUs by its peer, a transport entity may transmit an AK TPDU at any time.

The sequence number of an AK TPDU must not exceed the sequence number of the next expected DT TPDU, i.e. it must not be greater than the highest sequence number of a received DT TPDU, plus one.

A transport entity may send a duplicated AK TPDU containing the same sequence number, CDT, and subsequence number field at any time.

A transport entity may increase or decrease the upper window edge at any time.

A transport entity which receives an AK TPDU must consider the value of the YR-TU-NR field as its new lower window edge if it is greater than any previously received in a YR-TU-NR field, and the sum of YR-TU-NR and CDT as its new upper window edge subject to the procedures for sequencing AK TPDUs (discussed below).

A transport entity must not transmit or retransmit a DT TPDU with a sequence number outside the transmit window.

To allow a receiving transport entity to properly sequence a series of AK TPDUs that all contain the same sequence number and, thereby, use the correct CDT value, AK TPDUs may contain a subsequence parameter. For the purpose of determining the correct sequence of AK TPDUs, the absence of the subsequence parameter must be equivalent to the value of the parameter set to zero.

An AK TPDU is defined to be in sequence if:

a) The sequence number is greater than in any previously received AK TPDU, or

b) the sequence number is equal to the highest in any previously received AK TPDU, and the subsequence parameter is greater than in any previously received AK TPDU having the same value for YR-TU-NR field, or

c) the sequence number and subsequence parameter are both equal to the highest in any previously received AK TPDU and the credit field is greater than or equal to that in any previously received AK TPDU having the same YR-TU-NR field.

When the receiving transport entity receives an out of sequence AK TPDU, it is ignored.

For retransmission of AK TPDUs to establish window synchronization, a transport entity must not allow an interval W to pass without the transmission of an AK TPDU. If the transport entity is not using the procedure following setting CDT to zero (discussed below) or reduction of the upper window edge (also discussed below), and does not have to acknowledge receipt of any DT TPDU, then it must achieve this by retransmission of the most recent AK TPDU, with up-to-date window information.

To allow the receiving transport entity to process AK TPDUs in the correct sequence, as described above, the subsequence parameter must be included following

reduction of CDT. If the value of the subsequence number to be transmitted is zero, then the parameter should be omitted.

The value of the subsequence parameter must be zero (either explicitly or by absence of the parameter) if the sequence number is greater than the field in previous AK TPDUs, sent by the transport entity. If the sequence number is the same as the previous AK TPDU sent and the CDT field is equal to or greater than the CDT field in the previous AK TPDU sent, then the subsequence parameter must be equal to that in the previously sent AK TPDU. If the sequence number is the same as the previous AK TPDU sent and the CDT field is less than the value of the CDT field in the previous AK TPDU sent, then the subsequence parameter must be one greater than the value in the previous AK TPDU.

Due to the possibility of loss of AK TPDUs, the upper window edge as perceived by the transport entity transmitting an AK TPDU may differ from that perceived by the intended recipient. To avoid the possibility of deadlock, the retransmission procedure (section 4.10.1) should be followed for an AK TPDU, if it opens the transmit window after it was previously closed by sending an AK TPDU with CDT field set to zero.

The retransmission procedure terminates and the procedure for retransmission of AK TPDUs (to establish window synchronization) (discussed above) is used when:

a) An AK TPDU is received containing the flow control confirmation parameter, whose lower window edge and subsequence fields are equal to the sequence number and subsequence number in the retained AK TPDU;

b) An AK TPDU is transmitted with a sequence number higher than that in the retained AK TPDU, due to reception of a DT TPDU whose sequence number is equal to the lower window edge;

c) N transmissions of the retained AK TPDU have taken place. In this case, the transport entity must continue to transmit the AK TPDU at an interval of W.

An AK TPDU which is subject to the retransmission procedure must not contain the flow control confirmation parameter. If it is required to transmit this parameter

An AK TPDU which is subject to the retransmission procedure must not contain the flow control confirmation parameter. If it is required to transmit this parameter concurrently, an additional AK TPDU must be transmitted having the same values in the sequence, subsequence (if applicable) and credit fields.

The following procedure for retransmission of AK TPDUs after a transport entity has reduced the upper window edge (see "explicit flow control" discussed above) is used until the lower window edge exceeds the highest value of the upper window edge ever transmitted (i.e. the value existing at the time of credit reduction, unless a higher value is retained from a previous credit reduction). This retransmission procedure should be followed for any AK TPDU which increases the upper window edge, unless an AK TPDU has been received containing a flow control confirmation parameter, which corresponds to an AK TPDU transmitted following credit reduction, for which the sum of the credit and lower window edge fields (i.e. the upper window edge value) is greater than the lower window edge (YR-TU-NR field) of the transmitted AK TPDU.

The retransmission procedure for any particular AK TPDU must terminate when:

a) an AK TPDU is received containing the flow control confirmation parameter, whose lower window edge and subsequence fields are equal to the lower window edge and subsequence number in the retained AK TPDU, or

b) N transmissions of the retained AK TPDU have taken place. In this case, the transport entity must continue to transmit the AK TPDU at an interval of W.

NOTE

Retransmission of AK TPDUs is normally not necessary, except following explicit closing of the window (i.e. transmission of an AK TPDU with CDT field set to zero). If data is available to be transmitted, the retransmission procedure for DT TPDUs will ensure that an AK TPDU is received granting further credit where this is available. Following credit reduction, this may no longer be so, because retransmission may be inhibited by the credit reduction. The rules described in this section ensure that deadlock does not result in this case.

The rules for determining whether to apply the retransmission procedure to an AK TPDU may alternatively be expressed as follows. Let:

LWE = lower window edge
UWE = upper window edge
KUWE = lower bound on upper window edge held by remote transport entity

The retransmission procedure must be used whenever:

(UWE . LWE) and (KUWE = LWE)

i.e. when the window is opened and it is not known definitely that the remote transport entity is aware of this.

KUWE is maintained as follows. When credit is reduced, KUWE is set to LWE. Subsequently, it is increased only upon receipt of a valid flow control confirmation (i.e. one which matches the retained lower window edge and subsequence). In this case, KUWE is set to the implied upper window edge of the flow control confirmation, i.e. the sum of its lower window edge and the credit field. By this means, it can be ensured that KUWE is always less than or equal to the actual upper window edge in use by the transmitter of DT TPDUs.

At any time, an AK TPDU may be transmitted containing a flow control confirmation parameter. The lower window edge, subsequence and your credit fields must be set to the same value as the corresponding fields in the most recently received in-sequence AK TPDU.

An AK TPDU containing a flow control confirmation parameter should be transmitted whenever:

a) a duplicate AK TPDU is received with the value of YR-TU-NR, CDT, and subsequence fields equal to the most recently received AK TPDU, but not itself containing the flow control confirmation parameter;

b) an AK TPDU is received which increases the upper window edge but not the lower window edge, and the lower window edge and the upper window edge was formerly equal to the lower window edge; or

c) An AK TPDU is received which increases the upper window edge but not the lower window edge; and, the lower window edge is lower than the highest value of the upper window edge ever received (i.e., following credit reduction).

4.10.4 Release

There are no timers used only for release. The transport entity must use the following procedures:

a) explicit variant of normal release (see section 4.4.7);

b) frozen references (see section 4.4.18).

4.11 STRUCTURE AND ENCODING OF TPDUs

Table 4-8 specifies those TPDUs which are valid for each class and the code for each TPDU.

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

NOTE

When the encoding of a TPDU is represented using a diagram in this section, the following representation is used:

- Octets are shown with the lowest numbered octet to the left, higher numbered octets being further to the right;
- Within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

Table 4-8. TPDU Codes

TPDU	Validity Within Classes					see Section	Code
	0	1	2	3	4		
CR Connection Request	X	X	X	X	X	4.11.1	1110 xxxx
CC Connection Confirm	X	X	X	X	X	4.11.2	1101 xxxx
DR Disconnect Request	X	X	X	X	X	4.11.3	1000 0000
DC Disconnect Confirm		X	X	X	X	4.11.4	1100 0000
DT Data	X	X	X	X	X	4.11.5	1111 0000
ED Expedited Data		X	NF	X	X	4.11.6	0001 0000
AK Data Acknowledgement		NRC	NF	X	X	4.11.7	0110 zzzz
EA Expedited Data Acknowledgement		X	NF	X	X	4.11.8	0010 0000
RJ Reject		X		X		4.11.9	0101 zzzz
ER TPDU Error	X	X	X	X	X	4.11.10	0111 0000
not available (see note)						-	0000 0000
						-	0011 0000
						-	1001 xxxx
						-	1010 xxxx

KEY: xxxx (bits 4-1): used to signal the CDT
zzzz (bits 4-1): used to signal CDT in classes 2,3,4; set to 1111 in class 1
NF: Not available when the non-explicit flow control option is selected.
NRC: Not available when the receipt confirmation option is selected.

NOTE: These codes are already in use in related protocols defined by standards organizations other than CCITT/ISO.

Structure: All the transport protocol data units (TPDUs) must contain an integral number of octets. The octets in a TPDU are numbered starting from 1 and increasing in the order they are put into an NSDU. The bits in an octet are numbered from 1 to 8, where bit 1 is the low-ordered bit.

TPDUs contain, in the following order:

a) the header, comprising:

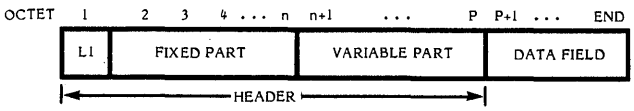
1) the length indicator (LI) field;

2) the fixed part;

3) the variable part, if present;

b) the data field, if present.

This structure is illustrated below:



Length Indicator Field: This field is contained in the first octet of the TPDUs. 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.

Fixed Part: The fixed part contains frequently occurring parameters including the code of the TPDU. The length and the structure of the fixed part are defined by the TPDU code and, in certain cases, by the protocol class and the formats in use (normal or extended).

NOTE

In general, the TPDU code defines unambiguously the fixed part. However, different variants may exist for the same TPDU code (see normal and extended formats).

TPDU Code: This field contains the TPDU code and is contained in octet 2 of the header. It is used to define the structure of the remaining header. The field is a full octet except in the following cases:

1110 xxxx	Connecting Request
1101 xxxx	Connection Confirm
0101 xxxx	Reject
0110 xxxx	Data Acknowledgement

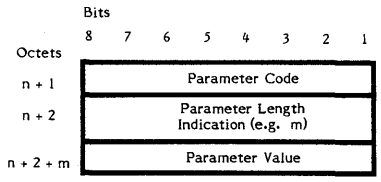
where xxxx (bits 4-1) is used to signal the CDT.

Only those codes defined in Table 4-8 are valid.

Variable Part: The variable part is used to define less frequently used parameters. If the variable part is present, it must contain one or more parameters. The number of parameters that may be contained in the variable part is indicated by the length of the variable part which is LI minus the length of the fixed part.

Since the currently defined minimum fixed part for headers that allow parameters is four octets, and since the value of length indicator field is limited to a maximum of 254, the maximum length of the variable part is 250 octets.

Each parameter contained within the variable part is coded as follows:



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 value, so the practical maximum number of different parameters is less. Parameter code 1111 1111 is reserved for possible extensions of the parameter code.

The parameter length indication indicates the length, in octets, of the parameter value field. The length is indicated by a binary number, m, with a theoretical maximum 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 248. 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 use bits 8 and 7 with the value 00.

Transport entities must accept the parameters defined in the variable part in any order. If any parameter is duplicated, then the later value will be used. A parameter not defined in this chapter must be treated as a protocol error in any received TPDU, except a CR TPDU. In a CR TPDU, an invalid parameter must be treated as a protocol error in class 0 and must be ignored in other classes.

All TPDU types may contain a checksum parameter (class 4 only) in their variable part. This parameter must always be present except when the non-use of checksum option is selected:

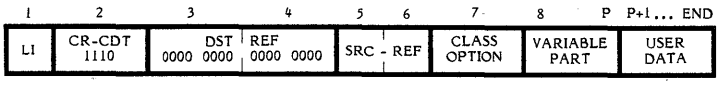
Parameter Code:	1100 0011
Parameter Length:	2
Parameter Value:	Result of checksum algorithm. This algorithm is specified in section 4.4.17.

Data Field: This field contains transparent user data. Restrictions on its size are noted for each TPDU.

4.11.1 Connection Request (CR) TPDU

The length of the TPDU must not exceed 128 octets.

Structure: The structure of the CR TPDU must be as follows:



LI: Described in section 4.11.

Fixed Part (Octets 2 to 7): The structure of this part must contain:

CR : Connection Request Code: 1110 (Bits 8-5 of octet 2);

CDT : Initial Credit Allocation (set to 0000 in classes 0 and 1 when specifie as preferred class) (Bits 4-1 of octet 2);

DST-REF: Set to zero;

SRC-REF: Reference selected by the transport entity initiating the CR TPDU identify the requested transport connection;

CLASS and OPTION : Bits 8-5 of octet 7 define the preferred Transport Protocol class to b operated over the requested Transport connection. This field ma take one of the following values:

0000	Class 0
0001	Class 1
0010	Class 2
0011	Class 3
0100	Class 4

The CR TPDU contains the first choice of class in the fixed part as above. Secon and subsequent choices are listed in the variable part, if required.

Bits 4-1 of octet 7 are reserved for options to be used on the requested transport connection.

The uses of bits 4-1 are as follows:

Bit	Option	
4	0	always
3	0	always
2	=0	use of normal formats in all classes
	=1	use of extended formats in classes 2,3,4
1	=0	use of explicit flow control in class 2
	=1	no use of explicit flow control in class 2

NOTE 1

The connection establishment procedure (see section 4.4.5) does not permit a given CR TPDU to request use of transport expedited data transfer service (additional option parameter) and no use of explicit flow control in class 2 (bit 1 = 1).

NOTE 2

Bits 4 to 1 are always zero in class 0 and have no meaning.

Variable Part (Octets 8 to p): The following parameters are permitted in the variable part:

Transport Service Access Point Identifier (TSAP-ID):

Parameter code : 1100 0001 for the identifier of the Calling TSAP.
1100 0010 for the identifier of the Called TSAP.

Parameter length : not defined in this standard.

Parameter value : identifier of the calling or called TSAP, respectively.

If a TSAP-ID is given in the request, it may be returned in the confirmation.

TPDU Size: This parameter defines the proposed maximum TPDU size (in octets including the header) to be used over the requested transport connection. The coding of this parameter is:

Parameter code : 1100 0000
Parameter length : 1 octet

Parameter values :

0000 1101 8192 octets (not allowed in class 0)
0000 1100 4096 octets (not allowed in class 0)
0000 1011 2048 octets
0000 1010 1024 octets
0000 1011 512 octets
0000 1000 256 octets
0000 0111 128 octets

Default value is 0000 0111 (128 octets)

Version Number (not used in class 0):

Parameter code : 1100 0100
Parameter length : 1 octet
Parameter value field: 0000 0001

Default value is 0000 0001 (not used in class 0)

Security Parameters (not used in class 0). This parameter is user defined:

Parameter code : 1100 0101
Parameter length : user defined
Parameter value : user defined

Checksum (not used in classes 0 through 3): This parameter must always be present in a CR TPDU requesting class 4, even if the checksum selection parameter is used to request non-use of the checksum facility.

Additional Option Selection (not used in class 0): This parameter defines the selection to be made as to whether or not additional options are to be used:

Parameter code : 1100 0110
Parameter length : 1
Parameter value :

Bit	Option
4	1= use of network expedited in class 1 0= non use of network expedited in class 1
3	1= use of receipt confirmation in class 1 0= use of explicit AK variant in class 1
2	0= checksums are to be used in class 4 1= checksums are not to be used in class 4
1	0= use of transport expedited data transfer service 0= no use of transport expedited data transfer service

Default value is 0000 0001

Bits related to options particular to a class are not meaningful if that class is not proposed, and may take any value.

Alternative Protocol Class (not used if class 0 is the preferred class):

Parameter code : 1100 0111
Parameter length : n

Parameter value is encoded as a sequence of single octets. Each octet is encoded as for octet 7 but with bits 4-1 set to zero (i.e. no alternative option selections permitted).

Acknowledge Time: This parameter conveys the maximum acknowledge time A_L to the remote transport entity. It is an indication only, and is not subject to negotiation (see section 4.10):

Parameter code	:	1000 0101
Parameter length	:	2
Parameter value	:	n, a binary number where n is the maximum acknowledge time, expressed in milliseconds.

Throughput:

Parameter code	:	1000 1001
Parameter length	:	12
Parameter value	:	
1st 3 octets	-	Target value, calling-called user direction
2nd 3 octets	-	Minimum acceptable, calling-called user direction
3rd 3 octets	-	Target value, called-calling user direction
4th 3 octets	-	Minimum acceptable, called-calling user direction

Values are expressed in octets per second.

Residual Error Rate:

Parameter code	:	1000 0110
Parameter length	:	3
Parameter value	:	
1st octet	-	Target value, power of 10
2nd octet	-	Minimum acceptable, power of 10
3rd octet	-	TSDU size of interest, expressed as a power of 2

Priority:

Parameter code	:	1000 0111
Parameter length	:	2

Parameter value : Integer (0 is the highest priority)

Transit Delay:

Parameter code : 1000 1000
Parameter length : 8
Parameter value :
1st 2 octets - Target value, calling-called user direction
2nd 2 octets - Maximum acceptable, calling-called user direction
3rd 2 octets - Target value, called-calling user direction
4th 2 octets - Maximum acceptable, called-calling user direction

Values are expressed in milliseconds.

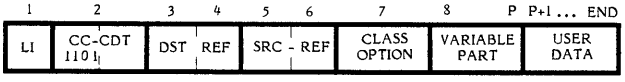
Reassignment Time: This parameter conveys the Time to Try Reassignment (TTR) which will be used when following the procedure for Reassignment after Failure (see section 4.4.12):

Parameter code : 1000 1010
Parameter length : 2
Parameter value : n, a binary number where n is the TTR value expressed in seconds.

User Data (Octets p + 1 to the end): No user data is permitted in class 0, and is optional in the other classes. Where permitted, it may not exceed 32 octets.

4.11.2 Connection Confirm (CC) TPDU

Structure: The structure of the CC TPDU must be as follows:



LI: Described in section 4.11.

Fixed Part (Octets 2 to 7): The fixed part must contain:

CC : Connection Confirm Code: 1101 (Bits 8-5 of octet 2);

CDT : Initial Credit Allocation (set to 0000 in classes 0 and 1) (Bits 4-1 of octet 2);

DST-REF : Reference identifying the requested transport connection at the remote transport entity;

SRC-REF : Reference selected by the transport entity initiating the CC TPDU to identify the confirmed transport connection;

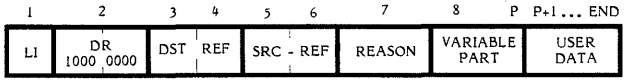
CLASS and OPTION : Defines the selected transport protocol class and option to be operated over the accepted transport connection according to the negotiation rules specified in section 4.4.5.

Variable Part (Octet 8 to p): The parameters are defined in section land are subject to the constraints stated in section 4.4.5 (connection establishment). Parameters ruled out by selection of an alternative class and option must not be present.

User Data (Octets p+1 to the end): The CC TPDU may contain a user data field subject to the constraints of the user data field of the CR TPDU (see section 4.10) and of the negotiation rules (see section 4.4.5).

4.11.3 Disconnect Request (DR) TPDU

Structure: The structure of the DR TPDU shall be as follows:



LI: Is described in section 4.11.

Fixed Part (Octets 2 to 7): The fixed part must contain:

DR : Disconnect Request Code: 1000 0000;

DST-REF : Reference identifying the transport connection at the remote transport entity;

SRC-REF : Reference identifying the transport connection at the transport entity initiating the command. Value zero when reference is unassigned;

REASON : Defines the reason for disconnecting the transport connection. This field must take one of the following values:

The following values may be used for classes 1 to 4:

1) 128 + 0 - Normal disconnect initiated by session entity
2) 128 + 1 - Remote transport entity congestion at connection request time
3) *128 + 2 - Connection negotiation failed (i.e. proposed class(es) not supported)
4) 128 + 3 - Duplicate connection detected
5) 128 + 4 - Mismatched references
6) 128 + 5 - Protocol error
7) 128 + 6 - Not used
8) 128 + 7 - Reference overflow
9) 128 + 8 - Connection request refused on this network connection
10) 128 + 9 - Not used
11) 128 + 10 - Header or parameter length invalid

The following values can be used for all classes:

12) 0 - Reason not specified

13) 1 - Congestion at TSAP

14) *2 - Session entity not attached to TSAP

15) *3 - Address unknown

NOTE

Reasons marked with an asterisk (*) may be reported to the TS-user as persistent, other reasons as transient.

Variable Part (Octets 8 to p): The variable part may contain:

a) A parameter allowing additional information related to the clearing of the connection.

Parameter code : 1110 0000

Parameter length : Any value provided that the length of the DR TPDU does not exceed the maximum agreed TPDU size or 128 when the DR TPDU is used during the connection refusal procedure

Parameter value : Additional information. This field is intended to be used by the transport entity for internal purposes

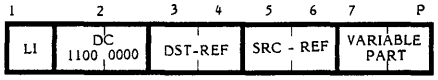
b) Checksum (see section 4.11).

User Data (Octets p+1 to the end): This field must not exceed 64 octets and is used to carry TS-user data. The successful transfer of this data is not guaranteed. When a DR TPDU is used in class 0, it must not contain this field.

4.10.4 Disconnect Confirm (DC) TPDU

This TPDU must not be used in class 0.

Structure: The structure of DC TPDU shall be as follows:



LI: Described in section 4.11.

Fixed Part (Octets 2 to 6): The fixed part must contain:

DC : Disconnect Confirm Code: 1100 0000;

DST-REF : See 4.11.2

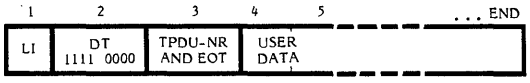
SRC-REF : See 4.11.2

Variable Part: The variable part may contain the checksum parameter (see section 4.11).

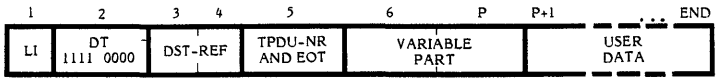
4.11.5 Data (DT) TPDU

Structure: Depending on the class and the option, the DT TPDU must have one of the following structures:

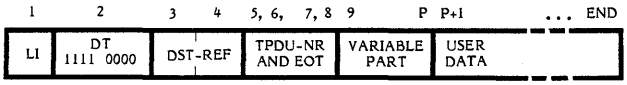
a) Normal format for classes 0 and 1



b) Normal format for classes 2, 3 and 4



c) Extended format for optional use in classes 2, 3 and 4



LI: Described in section 4.11.

Fixed Part: The fixed part must contain:

DT : Data Transfer Code: 1111 0000;

DST-REF : See subsection 4.11.2

EOT : When set to ONE, indicates that the current DT TPDU is the last data unit of a complete DT TPDU sequence (End of TSDU). EOT is bit 8 of octet 3 in class 0 and 1, bit 8 of octet 5 for normal formats for classes 2, 3 and 4 and bit 8 of octet 8 for extended formats;

TPDU-NR : TPDU send Sequence Number (zero in class 0); may take any value in class 2 without explicit flow control. TPDU-NR is bits 7-1 of octet 3 for classes 0 and 1, bits 7-1 of octet 5 for normal formats in classes 2, 3 and 4; octets 5, 6 and 7 together with bits 7-1 of octet 8 for extended formats.

NOTE

Depending on the class, the fixed part of the DT TPDU must use the following octets:

Classes 0 and 1: Octets 2 to 3;
Classes 2,3,4 normal format: Octets 2 to 5;
Classes 2,3,4 extended format: Octets 2 to 8.

Variable Part: The variable part may contain the checksum parameter (see section 4.11).

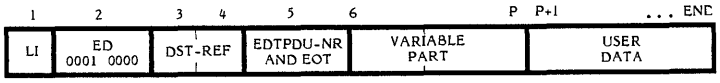
User Data Field: This field contains data of the TSDU being transmitted. The length of this field is limited to the negotiated TPDU size for this transport connection minus 3 octets in classes 0 and 1, and minus 5 octets (normal header format) or 8 octets (extended header format) in the other classes. The variable part, if present, may further reduce the size of the user data field.

4.11.6 Expedited Data (ED) TPDU

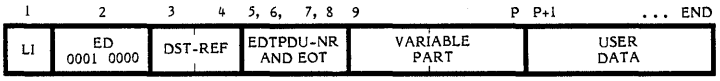
The ED TPDU must not be used in class 0 or in class 2 when the no explicit flow control option is selected or when the expedited data transfer service has not been selected for the connection.

Structure: Depending on the format negotiated at connection establishment, the ED TPDU must have one of the following structures:

a) Normal Format (classes 1, 2, 3, 4)



b) Extended Format (optional use in classes 2, 3, 4)



LI: Described in section 4.11.

Fixed Part: The fixed part must contain:

ED : Expedited Data code 0001 0000;

DST-REF : See section 4.11.2;

ED-TPDU-NR : Expedited TPDU identification number. ED-TPDU-NR is used in classes 1, 3 and 4 and may take any value in class 2. Bits 7-1 of octet 5 for normal formats and octets 5, 6 and 7 together with bits 7-1 of octet 8 for extended formats;

EOT : End of TSDU always set to 1 (bit 8 of octet 5 for normal formats and bits 8 of octet 8 for extended formats).

NOTE

Depending on the format, the fixed part must be either octets 2 to 5 or 2 to 8.

Variable Part: The variable part may contain the checksum parameter (see section 4.11).

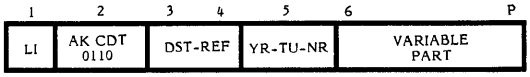
User Data Field: This field contains an expedited TSDU (1 to 16 octets).

4.11.7 Data Acknowledgement (AK) TPDU

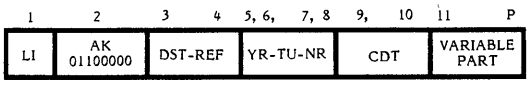
This TPDU must not be used for class 0 and class 2 when the "no explicit flow control" option is selected, not for class 1 when the network receipt confirmation option is selected.

Structure: Depending on the class and option agreed upon, the AK TPDU must have one of the following structures:

a) Normal Format (classes 1, 2, 3, 4)



b) Extended Format (optional use in classes 2, 3, 4)



LI: Defined in section 4.11.

Fixed Part: The fixed part must contain (in octet 2 to 5 when normal format is used, 2 to 10 otherwise) the following parameters:

AK : Acknowledgement code 0110;

CDT : Credit Value (set to 1111 in class 1). Bits 4-1 of octet 2 for normal formats and octets 9 and 10 for extended formats;

DST-REF : See section 4.11.2;

YR-TU-NR : Sequence number indicating the next expected DT TPDU number. For normal formats, bits 7-1 of octet 5 (bit 8 of octet 5 is not significant and must take the value 0). For extended formats, octets 5, 6 and 7 together with bits 7-1 of octet 8 (bit 8 of octet 8 is not significant and must take the value 0).

Variable Part: The variable part may contain the following parameters:

a) Checksum (see section 4.11);

b) Subsequence Number Class 4 only (optionally used).
This parameter is used to ensure that AK TPDUs are processed in the correct sequence. Its absence is equivalent to transmitting the parameter with a value of zero.
Parameter code : 1000 1010
Parameter length : 2
Parameter value : 16-bit sub-sequence number;

c) Flow Control Confirmation Class 4 only (optionally used).
This parameter contains a copy of the information received in an AK TPDU to allow the transmitter of the AK TPDU to be certain of the state of the receiving transport entity.
Parameter code : 1000 1011
Parameter length : 8
Parameter value : 64 bits, used as follows:

1. Lower Window Edge (32 bits)
Bit 32 is set to zero, bits 31 to 1 contain the YR-TU-NR value of the received AK TPDU. When normal format is in use, only the least significant seven bits (bits 1 to 7) of this field are significant.

2. Your Sub-Sequence (16 bits)
Contains the value of the sub-sequence parameter of the received AK TPDU, or zero if this parameter was not present.

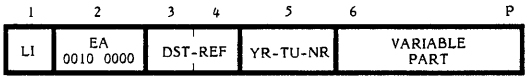
3. Your Credit (16 bits)
Contains the value of the CDT field of the received AK TPDU. When normal format is in use, only the least significant four bits (bits 1 to 4) of this field are significant.

4.11.8 Expedited Data Acknowledgement (EA) TPDU

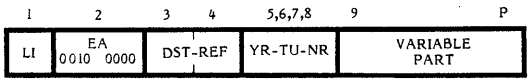
This TPDU must not be used for class 0 and class 2 when the no explicit flow control option is selected.

Structure: Depending on the option (normal or extended format) the TPDU structure must be:

a) Normal Format (classes 1, 2, 3, 4)



b) Extended Format (optional use in classes 2, 3, 4)



LI: Described in section 4.11.

Fixed Part: The fixed part must contain (in octets 2 to 5 when normal format is used, in octets 2 to 8 otherwise):

EA : Acknowledgement code 0010 0000;

DST-REF : See section 4.11.2;

YR-TU-NR : Identification of the ED TPDU being acknowledged. May take any value in class 2;

For normal formats, bits 7-1 of octet 5; bit 8 of octet 5 is not significant and must take the value 0. For extended formats, octets 5, 6 and 7 together with bits 7-1 of octet 8; bit 8 of octet 8 is not significant and must take the value 0.

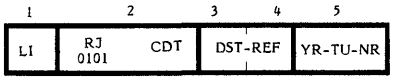
Variable Part: The variable part may contain the checksum parameter (see section 4.11).

4.11.9 Reject (RJ) TPDU

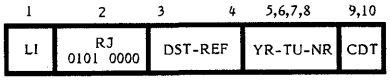
The RJ TPDU must not be used in classes 0, 2 and 4.

Structure: The RJ TPDU must have one of the following formats:

a) Normal Format (classes 1 and 3)



b) Extended Format (optional use in classes 1 and 3)



LI: Described in section 4.11.

Fixed Part: The fixed part must contain (in octets 2 to 5 when normal is used, in octets 2 to 10 otherwise):

RJ : Reject Code 0101 (Bits 8-5 of octet 2);

CDT : Credit Value (set to 1111 in class 1). Bits 4-1 of octet 2 for normal formats and octets 9 and 10 for extended formats;

DST-REF : See section 4.11.2;

YR-TU-NR : Sequence number indicating the next expected TPDU from which retransmission should occur.

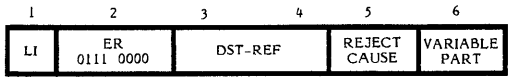
For normal formats, bits 7-1 of octet 5; bit 8 of octet 5 is not significant and must take the value 0. For extended formats, octets 5, 6 and 7 together with bits 7-1 of octet 8; bit 8 of octet 8 is not significant and must take the value 0.

Variable Part: There is no variable part for this TPDU type.

4.11.10 TPDU Error (ER)

This TPDU is defined as follows:

Structure:



LI: Described in section 4.11.

Fixed Part: The fixed part must contain:

ER : TPDU Error Code: 0111 0000;

DST-REF : See "Fixed Part" in section 4.11.2

REJECT CAUSE :	0000 0000	Reason not specified
	0000 0001	Invalid parameter code
	0000 0010	Invalid TPDU type
	0000 0011	Invalid parameter value.

Variable Part (Octets 6 to the end): The variable part may contain the following parameters:

a) Invalid TPDU

Parameter code 1100 0001
Parameter value:

Contains the bit pattern of the rejected TPDU up to and including the octet which caused the rejection. This parameter is mandatory in class 0.

b) Checksum (see section 4.11)

4.12 CONFORMANCE

This section is included to indicate what is required to conform to the ISO Standard.

A system claiming to implement the procedures specified by the ISO Standard must comply with the following requirements:

a) The system must implement class 0 or class 2 or both.

b) If the system implements class 3 or class 4, it must also implement class 2.

c) If the system implements class 1, it must also implement class 0.

d) For each class which the system claims to implement, the system must be capable of:

- initiating CR TPDUs or responding to CR TPDUs with CC TPDUs or both;
- responding to any other TPDU and operating network service in accordance with the procedures for the class;
- operating all the procedures for the class listed as mandatory in Table 4-9;
- operating those procedures for the class listed as optional in Table 4-9 for which conformance is claimed;
- handling all TPDUs of lengths up to the lesser value of:
  1) the maximum length for the class;
  2) the maximum for which conformance is claimed.

NOTE

This requirement indicates that TPDU sizes of 128 octets are always implemented.

Claims of conformance must state:

a) which class or classes of protocol are implemented;

b) whether the system is capable of initiating or responding to CR TPDUs or both;

c) which of the procedures listed as optional in Table 4-9 are implemented;

d) the maximum size of TPDU implemented; the value must be chosen from the following list and all values in the list which are less than this maximum must be implemented:

128, 256, 512, 1024, 2048, 4096 or 8192 octets.

Table 4-9. Provisions of Options

Procedure	Class 0	Class 1	Class 2	Class 3	Class 4
TPDU with checksum	not applicable	not applicable	not applicable	not applicable	mandatory
TPDU without checksum	mandatory	mandatory	mandatory	mandatory	optional
Expedited data transfer	not applicable	mandatory	mandatory	mandatory	mandatory
No expedited data transfer	mandatory	mandatory	mandatory	mandatory	mandatory
Flow control in class 2	not applicable	not applicable	mandatory	not applicable	not applicable
No flow control in class 2	not applicable	not applicable	optional	not applicable	not applicable
Normal formats	mandatory	mandatory	mandatory	mandatory	mandatory
Extended formats	not applicable	not applicable	optional	optional	optional
Use of receipt confirmation in class 1	not applicable	optional	not applicable	not applicable	not applicable
No use of receipt confirmation in class 1	not applicable	mandatory	not applicable	not appclicable	not applicable
Use of network expedited in class 1	not applicable	optional	not applicable	not applicable	not applicable
No use of network expedited in class 1	not applicable	mandatory	not applicable	not applicable	not applicable

4.13 STATE TABLES

The material in this section is an integral part of the body of the ISO Standard.

This section describes the transport protocol in terms of state tables. The state tables are given by showing the state of a transport connection, the events that occur in the protocol and the resultant state.

The following conventions are used:

a) the incoming events are shown in the tables using their abbreviate name defined in Table 4-10;

b) the states are shown in the tables using their abbreviate name defined in Table 4-11;

Table 4-10. Incoming Events

Abbreviate Name	Category	Name and Description
TCONreq	TS-user	T-CONNECT request primitive
TCONresp	TS-user	T-CONNECT response primitive
TDTreq	TS-user	T-DATA request primitive
TEXreq	TS-user	T-EXPEDITED DATA request primitive
TDISreq	TS-user	T-DISCONNECT request primitive
NDISind	NS-provider	N-DISCONNECT indication primitive
NCONconf	NS-provider	N-CONNECT confirm primitive
NRSTind	NS-provider	N-RESET indication primitive
CR	TPDU	Connect Request TPDU
CC	TPDU	Connect Confirm TPDU
DR	TPDU	Disconnect Request TPDU
DC	TPDU	Disconnect Confirm TPDU
AK	TPDU	Data Acknowledgement TPDU
EA	TPDU	Expedited Data Acknowledgement TPDU
DT	TPDU	Data TPDU
ED	TPDU	Expedited Data TPDU
ER	TPDU	Error TPDU
RJ	TPDU	Reject TPDU

c) the intersection of each state and event which is invalid is left blank. The action to be taken in this case is one of the following:

1) for an event related to the transport service (i.e. coming from the TS-user), take no action;

2) for an event related to a received TPDU, follow the procedure for treatment of protocol errors (see section 4.4.22) if the state of the supporting network connection makes it possible;

3) for an event falling into neither of the above categories (including those which are impossible by the definition of the behavior of the transport entity or NS-provider) take no action.

Table 4-11. States

Abbreviate Name	Name and Description
WFNC	Wait for network connection
WFCC	Wait for CC TPDU
WBCL	Wait before releasing (wait for CC TPDU before sending the DR TPDU)
OPEN	Transport connection is open
CLOSING	Release in progress
WFTRESP	Wait for T-CONNECT response
CLOSED	Transport connection is closed
WFCC-R	Wait for CC TPDU and reassignment in progress
WBCL-R	Wait before releasing and reassignment in progress
OPEN-R	Open and reassignment in progress
OPEN-WR	Open and wait for reassignment
CLOSING-R	Release in progress and reassignment in progress
CLOSING-WR	Release in progress and wait for reassignment
WFTRESP-WR	Wait for T-CONNECT response and wait for reassignment
WBCL-WR	Wait before releasing and wait for reassignment
WBOC	Wait before open complete (CC is unacknowledged)
WBOC-WR	Wait before open complete and wait for reassignment
CLOSING BOC	Wait before open complete and release in progress
CLOSING BOC-WR	Idem and wait for reassignment
AKWAIT	Waiting for acknowledgement of CC TPDU
REFWAIT	Waiting for frozen reference time
WFEA	Wait for EA TPDU
SCLOSEDW	Window is closed

d) at each intersection of state and event which is valid, the state tables specify an action which may include one of the following:

1) one action constituted of a list of any number of outgoing events (no, one or more) given by their abbreviated name defined in Table 4-12 followed by the abbreviate name of the resultant state (see Table 4-10);

2) several conditional actions separated by the symbol ; . Each conditional action contains a predicate followed by the symbol : and by an action as defined above. The predicates are boolean expressions given by their abbreviate name and defined in the clauses related to the state tables of each class. Only the action corresponding to the predicate which is true is to be taken.

Table 4-12: Outgoing Events

Abbreviate Name	Category	Name and Description
TCONind	TS-provider	T-CONNECT indication primitive
TCONconf	TS-provider	T-CONNECT confirm primitive
TDTind	TS-provider	T-DATA indication primitive
TEXind	TS-provider	T-EXPEDITED DATA indication primitive
TDISind	TS-provider	T-DISCONNECT indication primitive
NDISreq	NS-user	N-DISCONNECT request primitive
NRSTresp	NS-user	N-RESET response primitive
NCONreq	NS-user	N-Connect request primitive
CR	TPDU	Connect Request TPDU
CC	TPDU	Connect Confirm TPDU
DR	TPDU	Disconnect Request TPDU
DC	TPDU	Disconnect Confirm TPDU
AR	TPDU	Data Acknowledgement TPDU
EA	TPDU	Expedited Data Acknowledgement TPDU
DT	TPDU	Data TPDU
ED	TPDU	Expedited Data TPDU
ER	TPDU	Error TPDU
RJ	TPDU	Reject TPDU

e) the state tables may include the three following additional conventions:

1) informal comments giving explanatory materials;

2) references to notes using the following notation : (note number);

3) references to other actions defined in separate tables using the following notation : action number .

This annex includes the state table for class 0 and 2, the state table for classes 1 and 3, and the state table for class 4.

4.13.1 Incoming/Outgoing Events and State Tables

Table 4-10 specifies the names of the incoming events, classified as TS-user events, NS-provider events and TPDU events. Table 4-11 specifies the names of the states. Table 4-12 specifies the names of the outgoing events classified as TS-provider events, NS-user events and TPDU events.

4.13.2 State Tables For Classes 0 and 2

This section specifies the behavior of a transport entity for a transport connection of class 0 or class 2.

The description uses predicates defined in Table 4-13, and specific actions defined in Table 4-14.

The description does not include a complete specification of the data transfer but makes reference to the specification of the classes (see clause 8 and 10). Table 4-15 gives the state automata for classes 0 and 2.

Table 4-13. Predicates For Classes 0 and 2

Name	Description
P0	T-CONNECT request unacceptable
P1	Unacceptable CR TPDU
P2	No network connection available
P3	Network connnection available and open
P4	Network connection available and open in progress
P5	Class is class 0 (class selected in CC)
P6	Unacceptable CC
P7	Class is class 2
P8	Acceptable CC

Table 4-14. Specific Actions For Classes 0 and 2

Name	Description
(1)	If the network connection is not used by any other transport connection assigned to it, it may be disconnected
(2)	See 4.4.22 (receipt of an ER TPDU)
(3)	See data transfer procedures of the class
(4)	See expedited data transfer procedure of the class
(5)	An N-RESET response has to be issued once for the network connection if the network connnection has not been released

Table 4-15. State Table for Classes 0 and 2 (first part)

EVENT \ STATE	WFNC	WFCC	WBCL (Class 2 only)	OPEN	CLOSING (Class 2 only)	WFTRESP	CLOSED
TCONreq							P0: TDISind CLOSED; P2: NCONreq WFNC; P3: CR WFCC; P4: WFNC;
TCONresp						CC OPEN	
TDTreq				[3] OPEN			
TEXreq				[4] OPEN			
TDISreq	[1] CLOSED	P5:NDISreq CLOSED P7:WBCL;		P5:NDISreq CLOSED P7:DR CLOSING		DR CLOSED	
NCONconf	CR WFCC						
NRSTind	TDISind [1] [5] CLOSED	TDISind [1] [5] CLOSED	[1] [5] CLOSED	TDISind [1] [5] CLOSED	[1] [5] CLOSED	TDISind [1] [5] CLOSED	
NDISind	TDISind CLOSED	TDISind CLOSED	CLOSED	TDISind CLOSED	CLOSED	TDISind CLOSED	

Table 4-15. State Table for Classes 0 and 2 (second and last part) (Cont'd.)

STATE / EVENT	WFNC	WFCC	WBCL (Class 2 only)	OPEN	CLOSING (Class 2 only)	WFTRESP	CLOSED
CR							P1: DR(1) CLOSED; Not P1: TCONind WFTRESP
DR		TDISind [1] CLOSED	[1] CLOSED	P5: ; -------- P7:DC IDISind CLOSED	[1] CLOSED		DC CLOSED
DC	DOES NOT EXIST IN CLASS 0 (2)			P7: 1 CLOSED			CLOSED
CC		P8:TCOnconf OPEN P6 and P5: TDISind NDISreq CLOSED: P6 and P7: TDISind DR CLOSING	P5: (3) NDISreq CLOSED P7: DR CLOSING				DR CLOSED
AK	DOES NOT EXIST IN CLASS 0 (2)			[3] OPEN	CLOSING		CLOSED
EA	DOES NOT EXIST IN CLASS 0 (2)			[4] OPEN	CLOSING		CLOSED
ED	DOES NOT EXIST IN CLASS 0 (2)			[4] OPEN	CLOSING		CLOSED
DT				[3] OPEN	CLOSING		CLOSED
DR		TDISind [1] CLOSED	[1] CLOSED	[2]	[2]		CLOSED

NOTEs: 1) An ER TPDU must be sent in certain cases - see 4.3.6.
2) If received it must be processed as a protocol error - see 4.3.22.
3) A CR with class 2 has been sent and a CC class 0 is received.

4.13.3 State Tables For Classes 1 and 3

This clause specifies the behavior of a transport entity for a transport connection of class 1 or class 3.

The description uses the predicates defined in Table 4-16.

Specific actions are defined in Table 4-17 and specific additional notes are given in Table 4-18.

The description does not include a complete specification of the data transfer but makes reference to the specification of the classes. Table 4-19 gives the state automata for classes 1 and 3.

Table 4-16. Predicates For Classes 1 and 3

NAME	DESCRIPTION
PO	T-CONNECT request unacceptable
P1	T-CONNECT request acceptable and no available network connection can be used
P2	T-CONNECT request acceptable and a network connection can be used; the network connection opening is in progress
P3	T-CONNECT request acceptable and a network connection can be used; the network connection is open
P4	Recovery attempts may exceed the TTR delay (see note)
P5	Local choice
P6	Initiator of the transport connection
P7	Unacceptable CR TPDU

NOTE

It is possible that after a network disconnect and a successful reopening of the network connection, a new network disconnect indication is received prior to any valid TPDU. Therefore, this predicate may apply not only in the reassignment in progress states, but also in the normal states (i.e. WFCC, WBCL, OPEN, CLOSING).

Table 4-17. Specific Actions For Classes 1 and 3

NAME	DESCRIPTION
(1)	The network connection can be disconnected if not used by any transport connection assigned to it.
(2)	Reopen the network connection if not already requested for another multiplexed transport connection.
(3)	Network connection can be disconnected if not used by any transport connection and was locally opened.
(4)	Start TWR timer
(5)	Stop TWR timer
(6)	Issue an N-RESET response if not already done
(7)	See data transfer procedure for the class

Table 4-18. Specific Notes For Classes 1 and 3

NAME	DESCRIPTION
(1)	Any TPDU except DR and CC having an unknown destination reference
(2)	CC TPDU having an unknown destination reference or a mismatched source reference
(3)	CR TPDU which is not duplicated but rejected
(4)	Or send any DT or ED TPDU waiting for transmission or use N-DATA-ACKNOWLEDGE Request available and selected (class 1 only)
(5)	The TTR period starts at each transition to a state with reassignment in progress, if it has not been previously started and terminates when a valid TPDU is received for the transport connection
(6)	If the resultant state is CLOSED, the reference must be frozen, except in the cases described in section 4.4.18
(7)	An ER TPDU shall be sent in certain cases (see section 4.4.6)
(8)	Receipt of a DC TPDU is a protocol error since DC cannot be used for reassignment. It is recommended to stop the TWR timer (5) and to consider the transport connection as released (CLOSED STATE)

Table 4-18. Specific Notes For Classes 1 and 3 (Cont'd)

NAME	DESCRIPTION
(9)	Receipt of a TPDU other than CR, DR, DC or REJ in this state is a protocol error. It is recommended to stop the TWR timer (5), send a DR TPDU and enter the closing state
(10)	Or a DR with mismatched source reference has been received
(11)	Same action as for (9) and issue a TDISind

Table 4-19. State Table for Classes 1 and 3 (First Part - Connection-Responder Side)

STATE / EVENT	CLOSED	WFTRESP	WFTRESP -WR	WBCL -WR	WBOC	WBOC -WR	CLOSING BOC	CLOSING BOC-WR
TDISreq		DR CLOSED (6)	WBCL -WR		DR CLOSINGBOC	CLOSINGBOC -WR		
TCONresp		CC WBOC	WBOC-WR					
NRSTind		[4] [6] WFTRESP -WR	[6] WFTRESP -WR	[6] WBCL -WR	[4] [6] SBOC -WR	[6] WBOC -WR	[4] [6] CLOSING BOC-WR	[6] CLOSING BOC-WR
NDISIND		[4] WRTRESP -WR	WFTRESP -WR	WBCL -WR	[4] WBOC -WR	WBOC -WR	[4] CLOSING BPC-WR	CLOSING BOC-WR
CR	P7: DR(3,7) CLOSED (6) Not P7: TCONind WRFTRESP		[5] WRTRESP	[5] DR CLOSED (6)		[5] CC WBOC		DR [5] CLOSED (6)
DR	DC CLOSED				TDISind DC CLOSED (6)	DR [5] TDISind CLOSED (6)	CLOSED (6)	[5] DC CLOSED (6)
REJ	CLOSED				OPEN [7]	[5] [7] OPEN	CLOSING	[5] DR CLOSING
DC	CLOSED						CLOSED (6)	(8)
First TPDU other than CR, DR, DC or REJ	CLOSED			OPEN			CLOSING	(9)
TWR Time-Out			TDISind CLOSED (6)	CLOSED (6)		TDISind CLOSED (6)		CLOSED (6)

Table 4-19. State Table for Classes 1 and 3, Second Part - Connection-Initiator Side (Cont'd.)

EVENT \ STATE	CLOSED	WFNC	WFNC-R	WFCC	WFCC-R	WBCL	WBCL-R
TCONreq	P0: TDISind CLOSED P1: NCONreq WFNC P2: WFNC P3: CR WFCC						
NCONconf		CR WFCC	CR WFCC		CR WFCC		CR WBCL
NRSTind				CR [6] WFCC		CR [6] WBCL	
NDISind		[2] WFNC-R (5)	P4: TDISind CLOSED (6) Not P4: [2] WFNC-R;	P4: TDISind CLOSED (6) Not P4: [2] (5) WFCC-R;	P4: TDISind CLOSED (6) Not P4: [2] WFCC-R;	P4: CLOSED (6) Not P4: [2] (5) WBCL-R:	P4 or P5: CLOSED (6) Not(P4 or P5): [2] WBCL-R
TDISreq		[1] CLOSED (6)	[1] CLOSED (6)	WBCL	P5: CLOSED (6) Not P5: WBCL-R		
DR	(10) DC CLOSED			TDISind [1] CLOSED (6)		[1] CLOSED (6)	
CC				TCONconf AK(4) OPEN		DR CLOSING	
(1)	CLOSED						
(2)	DR CLOSED						

Table 4-19. State Table for Classes 1 and 3, Third Part (Open and Closing States) (Cont'd.)

STATE / EVENT	OPEN	OPEN-R	OPEN-WR	CLOSING	CLOSING-R	CLOSING-WR
NCONconf		REJ [7] OPEN			DR CLOSING	
TDISreq	DR CLOSING	CLOSING -R	CLOSING -WR			
NRSTind	REJ [7] [6] OPEN			DR CLOSING		
NDISind	P6 and P4: TDISind CLOSED (6) P6 and not P4: [2] (5) OPEN-R Not P6: [4] OPEN-WR	P4: TDISind CLOSED (6); Not P4: [2] OPEN-R		P6 and (P5 or P4: CLOSED (6) P6 and not (P4 or P5): [2] (5) CLOSING-R; Not P6: [4] CLOSING-WR	P4 or P5: CLOSED (6) Not (P4 or P5: [2] CLOSING	
REJ	REJ [7] OPEN		REJ [5,7] OPEN	CLOSING		DR [5] CLOSING
Time-Out TWR			TDISind (6) CLOSED			CLOSED (6)
DR	TDISind DC CLOSED (6)		TDISind DC [5] CLOSED (6)	CLOSED (6)		[5] CLOSED (6)
DC				[3] CLOSED (6)		(8)
DT, AK ED or EA TPDU	[7] OPEN		(11)	CLOSING		(9)

4.13.4 State Tables For Class 4

This section specifies the description of a class 4 transport connection.

Tables 4-20, 4-21, and 4-22 give the predicates, actions, and timer events for class 4, respectively.

Tables 4-23 and 4-24 are the state tables for a class 4 transport connection.

Table 4-20. Predicates For Class 4

NAME	DESCRIPTION
P0	Acceptable CR TPDU
P1	Acceptable CC TPDU
P2	Unduplicated ED TPDU
P3	New network connection required
P4	Retransmit counter , maximum
P5	Unacknowledged ED TPDU
P6	EOT is set to 1
P7	Sending credit = 0 (closed window)
P8	ED TPDU waiting for transmission

Table 4-21. Specific Actions For Class 4

NAME	DESCRIPTION
(1)	Set reference timer
(2)	Set/Reset window timer
(3)	Set/Reset Inactivity timer
(4)	Set/Reset Retransmit timer and set count = count + 1
(5)	Reduce credit/window
(6)	Update credit/window
(7)	Transmit DT TPDUs stored up to upper window

Table 4-22. Timer Events For Class 4

NAME	DESCRIPTION
Retrans-t	Retransmission timer
Ref-t	Reference timer
I-t	Inactivity timer
W-t	Window timer

Table 4-23. Class 4 Connection/Disconnection

STATE / EVENT	CLOSED	WFNC	WBCL	WFCC	WFTRESP	AKWAIT	OPEN	CLOSING	REFWAIT
CR	PO: TCONind WFTRESP NOT PO: DR CLOSED:				WFTRESP	AKWAIT	OPEN	CLOSING	REFWAIT
DR				TDISind [1] REFWAIT	TDISind DC [1] REFWAIT	TDISind DC [1] REFWAIT	TDISind DC [1] REFWAIT	[1] REFWAIT	DC REFWAIT
DC								REFWAIT	REFWAIT
CC				NOT P1: TDISind DR [4] CLOSING P1: TCONCONF [3] AK OPEN			AK or DT or ED [3] OPEN	DR CLOSING	REFWAIT
AK						[3] [2] OPEN	See Table 4-24 [3] OPEN	CLOSING	REFWAIT
ED						TEXind [4] [3] EA→ OPEN	P2: TEXind [3] EA OPEN; Not P2: [3] EA OPEN	CLOSING	REFWAIT
EA							See Table 4-24 OPEN	CLOSING	REFWAIT
DT							See Table 4-24 OPEN	CLOSING	REFWAIT

Table 4-23. Class 4 Connection/Disconnection (Cont'd.)

STATE / EVENT	CLOSED	WFNC	WBCL	WFCC	WFTRESP	AKWAIT	OPEN	CLOSING	REFWAIT
ER			[1] REFWAIT	TDISind [1] REFWAIT	TDISind [1] REFWAIT	TDISind DR [4] CLOSING	TDISind DR [4] CLOSING	[1] REFWAIT	REFWAIT
TCINreq	P3:NCONreq WFNC; Not P3: CR [4] WFCC								
TCONresp					CC [4] AKWAIT				
TDISreq				[5] WBCL	DR [1] REFWAIT	DR [4] CLOSING	DR [4] CLOSING		
NDISind		TDISind CLOSED		TDISind [1] REFWAIT	TDISind [1] REFWAIT	TDISind [1] REFWAIT	TDISind [1] REFWAIT	[1] REFWAIT	REFWAIT
NRSTind	CLOSED	WFNC	WBCL	WFCC	WFTRESP	AKWAIT	OPEN	CLOSING	REFWAIT
TDT						BUILD DT TPDU STORE DT TPDU AKWAIT	See Table 4-24 OPEN		
NCONconf		CR [4] WFCC							

Table 4-23. Class 4 Connection/Disconnection (Cont'd.)

STATE / EVENT	CLOSED	WFNC	WBCL	WFCC	WFT-RESP	AKWAIT	OPEN	CLOSING	REFWAIT
Ref-t									CLOSED
Retrans-t				P4:CR [4] WFCC Not P4: DR [4] TDISind CLOSING		P4:CC [4] AKWAIT Not P4: DR [4] TDISind CLOSING	P4:OPEN See Table 4-24 Not P4: DR [4] TDISind CLOSING	P4:DR [4] CLOSING Not P4: [1] REFWAIT	
I-t							TDISind [4] CLOSING		
TEXreq							P5:STORE ED TPDU OPEN Not P5: ED [4] OPEN		

Table 4-24. Class 4 Data Transfer

EVENT \ STATE	OPEN	WFEA	SCLOSEDW
DT	P6: TDTind [2] AK [5] [3] OPEN Not P6: AK [5] [3] [2] OPEN	P6: TDTind [2] AK [5] [3] WFEA Not P6: [5] [3] [2] WFEA	P6: TDTind [2] AK [5] [3] SCLOSEDW Not P6: [5] [3] [2] SCLOSEDW
AK	[6] [3] P7: SCLOSEDW Not P7: OPEN	[6] WFEA	Not P7: [7] OPEN; P7: SCLOSEDW
EA		P8: ED [4] [3] WFEA; Not P8 and P7: OPEN Not P7: SCLOSEDW	SCLOSEDW
TDTreq	[7] Not P7: OPEN P7: SCLOSEDW	STORE DT TPDU WFEA	STORE DT TPDU SCLOSEDW
TEXreq	ED [4] [3] WFEA	STORE ED TPDU WFEA	ED [4] [3] WFEA
W-t	AK [2] OPEN	AK [2] WFEA	AK [2] SCLOSEDW

4.14 CHECKSUM ALGORITHMS

This section is provided as information for implementors and is not an integral part of the body of the ISO Standard.

4.14.1 Symbols

The following symbols are used:

CO variables used in the algorithms
C1

i number (i.e. position) of an octet within the TPDU (see 4.10)

n number (i.e. position) of the first octet of the checksum parameter

L length of the complete TPDU

X value of the first octet of the checksum parameter

Y value of the second octet of the checksum parameter.

4.14.2 Arithmetic Conventions

Addition is performed in one of the two following modes:

a) modulo 255 arithmetic;

b) one's complement arithmetic in which if any of the variables has the value minus zero (i.e. 255), it shall be regarded as though it were plus zero (i.e. 0).

4.14.3 Algorithm For Generating Checksum Parameters

1) Set up the complete TPDU with the value of the checksum parameter field set to zero.

2) Initialize C0 and C1 to zero.

3) Process each octet sequentially from i = 1 to L by

   a) adding the value of the octet to C0, then

   b) adding the value of C0 to C1.

4) Calculate X and Y such that

$$X = -C1 + (L-n).C0 - (L-n-1)$$
$$Y = C1 - (L-n+1).C0 + (L-n)$$

5) Place the values X and Y in octets n and (n + 1), respectively.

NOTE

This algorithm calculates

which is equal to zero, if the formulas in section 4.3.17 are followed, since

4.14.4 Algorithm For Checking Checksum Parameters

1) Initialize C0 and C1 to zero.

2) Process each octet of the TPDU sequentially from i = 1 to L by:

   a) adding the value of the octet to C0; then

   b) adding the value of C0 to C1.

3) If, when all the octets have been processed, either or both of C0 and C1 does not have the value zero, the checksum formulas in section 4.4.17 have not been satisfied.

NOTE

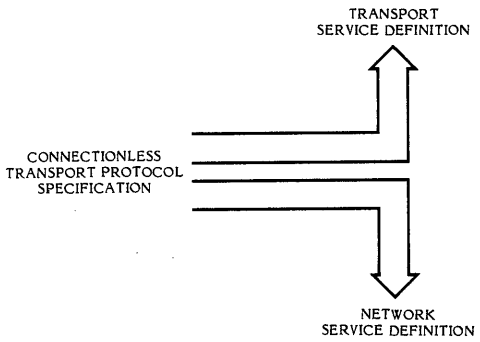
The nature of the algorithm is such that it is not necessary to compare explicitly the stored checksum bytes.

CHAPTER 5
TRANSPORT LAYER (CONNECTIONLESS)

5.1 INTRODUCTION

The Transport Protocol Standard (ISO DP 8073) is one of a set of International Standards produced to facilitate the interconnection of computer systems. The set of standards covers the services and protocols required to achieve such interconnection.

The Transport Protocol Standard is positioned with respect to other related standards by the layers defined in the Reference Model for Open Systems Interconnection (ISO 7498). In particular, it is a protocol of the Transport Layer. It is most closely related to, and lies within the field of application of the Transport Service Definition (DP 8072) and the Working Draft for an Addendum to the Transport Service Definition Covering Connectionless Data Transmission (SC16 N1445), the Network Service Definition (DP8438), and the Addendum to the Network Service Definition Covering Connectionless Data Transmission (TC97/SC6/N2728). The interrelationship of these standards is depicted below:



A document (ISO/TC 97/SC16N, February 1983) entitled "Working Draft for an addendum to the Transport Protocol Specification Covering Connectionless Data Transmission" Specifies transport architecture for connectionless systems. It is the specification followed by release 1 of the iNA960 and the source of this section.

5.1.1 Scope and Field of Application

This chapter specifies:

- procedures for connectionless transmission of data and protocol control information from one transport-entity to a peer transport-entity;
- the encoding of the Transport Protocol Data Units used for the transmission of data and control information;
- procedures for the correct interpretation of Transport protocol control information and
- the functional requirements for implementations claiming conformance to the ISO Standard

The procedures are defined in terms of:

- the interactions among peer transport-entities through the exchange of Transport Protocol Data Units;
- the interactions between a transport-entity and a Transport Service user through the exchange of Transport Service primitives; and
- the interactions between a transport-entity and a Network Service provider through the exchange of Network Service primitives.

This specifies a connectionless Transport Protocol. A connection-oriented Transport Protocol is specified in Chapter 4 of this manual

5.1.2 Reference Documents

The following are the sources containing information relevant to the definition of transport layer architecture for connectionless systems:

ISO 7498 Information Processing Systems - Open Systems Interconnection - Basic Reference Model

SC16 N1194 Addendum to the Reference Model Covering Connectionless Data Transmission

DP 8073 Information Processing Systems - Open Systems Interconnection - Transport Protocol Specification

DP 8072 Information Processing Systems - Open Systems Interconnection - Transport Service Definition

SC 16 N 1445 Working Draft for an Addendum to the Transport Service Definition Covering Connectionless Data Transmission

DP 8348 Information Processing Systems - Data Communications - Network Service Definition

SC6 N 2728 Addendum to the Network Service Definition Covering Connectionless Data Transmission

5.1.3 Definitions of Special Terms and Abbreviations used in this Chapter

The following are special terms and abbreviations used in this section:

Source-transportation-address:
Identifies the Transport Service user that acts as the source of data during a particular instance of Transport-connectionless data transmission.

Destination-transport-address:

Identifies the Transport Service user that acts as the sink of data during a particular instance of Transport-connectionless data transmission.

Data Units:

TPDU - Transport Protocol Data Unit
TSDU - Transport Service Data Unit
TSDU - Network Service Data Unit

Transport Protocol Data Units:

UD TPDU Unit data TPDU

TPDU Fields:

LI - Length Indicator
CDT - Credit

Parameters:

Source TSAP-id - Source Transport Service Address Parameter-identifies
Destination TSAP-id - Destination Transport Service Address Parameter-identifier

Miscellaneous:

TS-user: Transport Service User
TSAP: Transport Service Address Parameter
NSAP: Network Service Address Parameter

5.2 OVERVIEW OF THE TRANSPORT PROTOCOL

The following provides an overview of the service it provides and the services it expects from the Network Layer and the SSO model.

5.2.1 Service Provided by the Transport Layer

The service provided by the protocol herein described is a connectionless Transport Service. The connectionless Transport Service is described in the Addendum to the

Transport Service Definition Covering Connectionless Data Transmission (SC 16N)
The Transport Service Primitives provided are summarized below:

Primitive	Parameters
T_Unit_Data Request	To transport address, from-transport address quality of service TS-User Data
Indication	To-transport-address, from-transport address TS-User Data. Checksums Result (values are passed, failed or not checked)

5.2.2 Service Assumed from the Network Layer

The Transport protocol described in this chapter can operated over the connection mode network service defined in SC6 N2744 and over the connectionless mode network service defined in SC6 N2611R. When operating over a connection mode network service, the following network service primitives are used.

Primitive	Parameters
N_CONNECT.request	Called Address, Calling Address, Quality of Service
N_CONNECT.indication	Called Address, Calling Address
N_CONNECT.response	
N_CONNECT.confirm	
N_DATA.request	NS_User_Data
N_Data.indication	NS_User_Data

Primitive	Parameters
N_DISCONNECT request	
N_DISCONNECT indication	Reason

When operating over a connectionless mode network service, the following network service primitives are used:

Primitive	Parameters
N_UNITDATA Request	Destination Address Source Address Quality of Service NS_User_Data
N_UNITDATA Indication	Destination Address Source Address NS_User_Data

NOTE

Only those parameters pertinent to this (connectionless) transport protocol are listed above.

5.3 FUNCTIONS OF THE TRANSPORT LAYER

The following provides a overview of the Transport Layer in terms of the functions performed by that layer.

5.3.1 Connectionless Data Transfer Functions

The purpose of connectionless data transfer is to allow the transfer of data between corresponding TS-Users on a connectionless basis. This service provides for single-access data transfer for corresponding TS-Users without the overhead of transport connection establishment and termination. These functions are primarily

intended to benefit those applications where the reliability of data delivery provided by the transport protocol is not important because the applications involve sufficient data transmission redundancy or because the application protocol ensure the level of reliability required for its own operation.

5.3.2 Overview of Functions

The functions in the Transport layer are at least those necessary to bridge the gap between the service available from the Network Layer and the service to be offered to the Transport users.

The functions in the Transport Layer are concerned with the enhancement of the quality of service, including all aspects of cost optimization. The functions are:

- Network Service Selection,
- Address Mapping, and
- TSDU Delimiting

The first of these (Network Service Selection) selects the Network Service that best matches the requirement of the TS-User, taking into account charges for various services.

Address Mapping determines the Network address that is used as the destination-address parameter in an N_Unitdata Request or as the called address parameter in an N_CONNECT Request by examining the Transport address specified by the Destination-TSAP address parameter of a T_Unitdata Request.

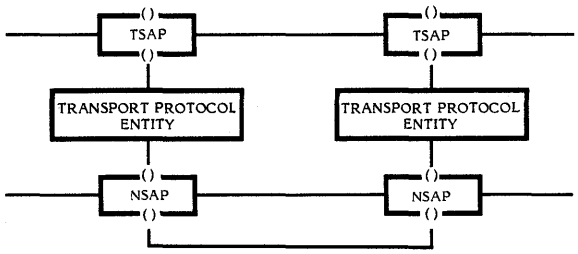
NOTE

The way in which this mapping is performed is determined by Transport Layer Management.

TDSU delimiting determines the beginning and end of a TSDU

5.3.3 Model of the Transport Layer

A transport protocol entity communicates with a TS-User through one or more TSAPs by means of Transport Service primitives (defined in DP 8072 and the Addendum to the Transport Service Definition Covering Connectionless Data Transmission). These Transport Service primitives cause or a result from the exchange of TPDUs between peer transport-entities engaged in connectionless data transmission or supporting a transport-connection. These protocol exchanges are effected by making use of the services of the Network Layer, as defined in the Network Service Definition and in the Addendum to the Network Service Definition Covering Connectionless Data Transmission, through one or more NSAPs. The following illustrates these relationships:



The model of Transport connectionless data transmission is presented in Clause 9.3 of the Addendum to the Transport Service Definition Covering Connectionless Data Transmission.

5.4 PROTOCOL MECHANISMS AND PROCEDURES

The protocol mechanisms and procedures described below are in addition to those described in Chapter 4 and in DP8073.

5.4.1 Transport Protocol Data Unit (TPDU) Transfer

The purpose of this mechanism is to convey transport protocol data units in user data fields of network service primitives.

The applicable network service primitives are:

- N_DATA (Request, Indication)
- N_UNITDATA (Request, Indication)

The name of the TPDU defined for the connectionless protocol is UNIT DATA (abbreviated UD).

5.4.2 Connectionless Data Transfer Over Connectionless Mode Network Service

The purpose of this mechanism is one time, one-way transfer of a TSDU between TSUsers without confirmation of receipt, without transport connection establishment and release, and without network connection establishment and release.

Network service primitives involved are:

N_UNITDATA (Request, Indication)

TPDUs and fields used are:

TPDU	Fields
UD	Checksum (optional)
	Source TSAP-ID
	Destination TSAP-ID
	User Data

Procedures for sending a UD TPDU in this case are as follows:

- The source and destination address parameters of the T_UNITDATA request service primitives are used to determine the source network address, source TSAP-ID, destination network address, and destination TSAP-ID.

- The quality of service parameter in the T_UNITDATA request is used to determine if a checksum should be included in the Unit Data (UD TPDU).

- A UD TPDU is constructed with a checksum parameter (optional), a source TSAP-ID, a destination TSAP-ID, and the User Data field from the T_UNITDATA request.

- An N_UNITDATA request service primitive is issued with the source and destination network addresses determined above, the quality of service from the T_UNITDATA request and a user data field containing the UD TPDU.

Procedure for UD TPDU are as follows:

- The UD TPDU arrives in the user data field of an N_UNITDATA indication. The transport entity constructs a T_UNITDATA indication and provide it to the appropriate transport user.

- The source network address from the N_UNITDATA indication and the source TSAP_ID from UD TPDU are used to determine the source address parameter for the T_UNITDATA indication. The destination network address from the N_UNITDATA indication and the destination TSAP_ID from the UD TPDU are used to determine the destination address parameter for the T_UNITDATA indication.

- If a checksum result parameter is present in the UD, a checksum verification of the UD is made using the algorithm defined for the transport protocol in Chapter 4 (and DP8073). The result of the checksum verification is passed in the checksum result parameter of the T_UNITDATA indication (passed, failed, or not checked).

- The user data field UD is mapped to the user data parameter of the T_UNITDATA.indication.

5.4.3 Connectionless Data Transfer Over Connection Mode Network Service

The purpose of this mechanism is one-time, one-way transfer of a TSDU between TS-Users without confirmation of receipt, without transport connection establishment and release, and with network connection establishment and release.

Network service primitives involved are:

N_CONNECT (Request, Indication, Response, Confirm)
N_DATA (Request, Indication)
N_DISCONNECT (Request, Indication)

TPDUs and fields used are:

TPDU	Fields
UD	Checksum (Optional)
	Source TSAP-ID
	Destinations TSAP-ID
	User Data

Procedures for sending a UD TPDU in this case are as follows:

- The source and destination address parameters of the T_UNIT DATA request primitive are used to determine the source network address, source TSAP-ID, destination network address, and destination TSAP-ID.

- The quality of service parameter in the T_UNIT DATA request is used to determine if a checksum should be included in the Unit Data (UD) TPDU.

- If a network connection to the destination network address does not already exist, an N_CONNECT request is issued. If an N_DISCONNECT indication is received in response, the procedure terminates.

- If an N_CONNECT confirm is received, a UD TPDU is constructed with a checksum parameter (optional), a source TSAP-ID, a destination TSAP-ID, and

the user data field from the T_UNITDATA request. An N_DATA request service primitive is issued with the UD TPDU contained in the user data field.

- Those transport classes that use timers set a timer to await an N_DISCONNECT indication.

- When an N_DISCONNECT indication is received the procedure terminates. If the timer set to await an N_DISCONNECT.indication expires and the associated network connection is not multiplexed, an N_DISCONNECT request is issued and the procedure terminates. If the network connection is multiplexed when the timer expires, the procedure terminates.

Procedures for receiving a UD TPDU are as follows:

- In order to receive a UD TPDU, an NCONNECT indication may have been received and an NCONNECT Response issued to establish a network connection between the correspondent TS-entities.

- The UD TPDU arrives in the user data field of an N_DATA indication. The transport entity constructs a T_UNITDATA indication and provides it to the appropriate transport user.

- The source address parameter for the T_UNITDATA indication is determined from the remote network address associated with the network connection and the source TSAP_ID from the UD TPDU. The destination address parameter for the T_UNITDATA indication is determined from the local network address associated with the network connection and the destination TSAP_ID from the UD TPDU.

- If a checksum parameter is present in the UD, a checksum verification of the UD is made using the algorithm defined for the transport protocol in Chapter 4 (and DP8073). The result of the checksum verification is passed in the checksum result parameter of the T_UNITDATA INDICATION (passed, failed or not checked).

- The user data field of the UD is mapped to the user data parameter of the T_UNITDATA indication. If the network connection over which the UD TPDU arrived is not multiplexed, an N_DISCONNECT request is issued. In any case, the procecdure terminates.

5.5 PROTOCOL CLASSES

Connectionless mode of operation introduces no additional classes to the transport protocol. Provision of this mode of operation is optional within all classes.

5.6 ENCODING OF THE UNIT DATA (UD TPDU)

The procedures described in this section require one new TPDU. The encoding of that TPDU, Unit Data (UD), is described below.

As in other TPDU's, a UD TPDU contains an integral number of octets. The octets in a TPDU are numbered starting from 1 and increasing in the order they are put into an NSDU. The bits in an octet are numbered from 1 to 8, where bit 1 is the low-order bit.

TPDUs contain (in the following order):

(1) The header, comprising:

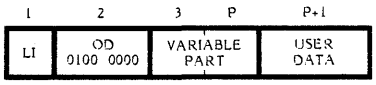
- The length indicator (LI) field;
- The fixed part;
- The variable part, if applicable;

(2) The data field, if applicable.

The length indicator field the fixed part and the variable part are as described in section 4.3.2 of Chapter 4 (or in DP 8073).

5.7 UNIT DATA (UD) TPDU STRUCTURE

The following illustrates the structure of the UD TPDU.



The several octets of this structure contain the following:

- Length Indicator (LI) - - Octet 1
- Fixed Part = 0100 0000 - - Octet 2
- Variable Part - - Octets 8 through P
- User Data - - Octet P + 1

The following parameters are permitted in the variable part:

a) Transport Service Access Point Identifier (TSAP-ID)

Each UD TPDU contains Source and Destination TSAP-identifiers in the variable part of its header as follows:

Parameter Code	Source TSAP 11000001 Destination TSAP 11000010
Parameter length	not defined
Parameter value	identifer of the source or Destination TSAP, respectively.

b) Checksum

A UD TPDU may optionally contain the checksum parameter in the variable part of the header:

Parameter Code	11000011
Parameter Length	2
Parameter Value	result of the checksum algorithm

The User Data contains the data of the TSDU being transmitted. The length of this field is unrestricted.

5.8 CONFORMANCE

The conformance information in Chapter 4 (or in DP 8073) is applicable. In addition, implementation claiming conformance must specify whether or not the connectionless data transmission option is supported.

CHAPTER 6
NETWORK MANAGEMENT

6.1 INTRODUCTION

The Network Management "Layer" supplies a network with planning, operation and maintenance facilities. (Network management is actually implemented as a function distributed over "other" layers and for this reason, it is treated as a "facility".) The planning capability gathers network usage information to help the user determine when to expand the network. Operation deals with normal, day to day network functions, such as initialization, termination, monitoring and performance optimization. Maintenance deals with detection, isolation, amputation and repair of network faults. Many functions can be performed both on local and remote nodes.

The Network Management Facility has an interface to every network layer in the architecture. Each layer provides the NMF with an interface in which the NMF can access the layer's internal database.

The NMF can perform operations on remote nodes. To do this, it uses the services provided by the "other" layers (Transport, Network and Data Link Layers) to communicate with the NMF on the remote node, which performs the desired operation at that node.

6.1.1 Purpose of This Section

This is an architectural specification, not a NMF implementation specification. The purpose of this chapter is to describe the essential functions, interfaces and algorithms which are necessary to implement a NMF module which will execute properly on the various layers at the local node and communicate properly with the NMF residing on remote nodes. This document does not describe or mandate a particular implementation strategy, since communication software can be implemented in a wide variety of execution environment.

6.2 GOALS AND NON-GOALS OF NMF ARCHITECTURAL DESIGN

The goals of the NMF design are:

Planning:

Capture usage statistics to help plan network expansion. Statistics on the various layers will be maintained by the layer itself and will be available to users via an interface with the NMF.

Operation:

The NMF should capture enough information to evaluate the settings of parameters affecting network performance. The NMF provides facilities to inspect and adjust these parameters to determine their effect on network performance. This allows users to monitor the operation of each layer.

Maintenance:

Part of the 'maintenance' goals of the NMF are present above in the section dealing with operation monitoring. In addition, the NMF provides features to determine the presence of hosts and the viability of their connection to the network. A facility to down-line dump remote systems is present. Error conditions or Events occurring in each layer are noted by the NMF Event Logger and can be displayed at the local and remote nodes.

Distributed Management:

Whatever facilities the NMF provides on the local node, the NMF can provide the same facility on a remote node. Enquiries into the state of the network can be made from any system on the network. There is no concept of a central network control station. Network management is the responsibility of every station.

Initialization:

NMF provides initialization and remote loading facilities. This allows systems with no local mass storage to be booted by a remote node. The facility provided is general enough to service a wide variety of systems.

The NMF has the following non-goals:

Security:

NMF does not support the concept of a super user. The facility available to all users is the same. NMF does not provide safe-guards or protection against malicious users.

6.3 FUNCTIONAL OVERVIEW

The Network Management Facility supplies a network with planning, operation and maintenance facilities. The planning capability gathers network usage information to help the user determine when to expand the network. Operation deals with normal, day to day network functions, such as initialization, termination, monitoring and performance optimization. Maintenance deals with detection, isolation, amputation and repair of network faults. Many functions can be performed both on local and remote nodes.

The functions needed to implement the planning, operation and maintenance goals overlap considerably. For instance, both planning and maintenance need access to the layer data bases, so the data base access functions are common to both. Similarly, operation and maintenance share a requirement to be able to remotely bootstrap a node. Therefore, the functions provided by the Network Management Facility are divided into groups of similar functions instead of being grouped according to their respective goals.

The NMF provides layer management, echo testing, limited debugging facilities and the ability to down-line load and up-line dump a remote system.

Layer management deals with manipulating the internal data base of a layer. NMF can examine and modify network counters of lower layers that indicate how the network is performing. It can examine and change the value of network parameters. Data base operations can be performed on the local node or a remote node. If performed on a remote node, the NMF uses the Transport Layer to create a virtual connection to the NMF in the remote node. The NMF in the remote node then performs the desired function.

As mentioned above, an echo facility is provided. Using this facility, the host can determine if a node is present on the network or not, test the communication path to that node and determine the functionality of the remote node. NMF transmits a packet to the remote node, then listens for that node to echo it.

The NMF provides the host with the ability to read or set memory of any host present on the network. This feature is provided as an aid to debugging. In the case of an operation to be performed on a remote node, the NMF transmits a message via the local Transport Layer to the NMF residing on the remote host. The remote NMF carries out the command.

NMF has the facility to record events which reflect a problem with the communication hardware or software. When an error occurs, the event, along with relevant information, is written into a log file. NMF has the facility to record events that occur locally as well as those that occur on remote nodes. Recording events may help to maintain the network, recover from failures, and plan for the future.

NMF can down-line load any system present on the network. A simple data link level protocol is used to ensure reliability. This facility can be used to load data bases; to boot systems without local mass storage; to boot a set of nodes remotely, thus ensuring that they have the same version of software; etc.

Up-line dumping is somewhat similar to down-line loading. Up-line dumping can be initiated by a remote node only. The remote node issues a 'dump command' and the target node responds with a dump response packet containing the memory image requested. In this case too, the target node utilizes the raw data link facilities.

6.4 LAYER MANAGEMENT

The NMF needs access to the data base of all other layers so that it can provide planning and maintenance aids. The element of a layer's internal data base is termed an object. The NMF can read, read and clear, and set an object in a layer's data base.

There are four types of objects; Parameters, Counters, Statistics and Values. Parameters adjust layer operation. They may be read and set, and change value only through the set operation. Counters record the number of times an event occurs. They are unsigned integers which either wrap around to zero on overflow or 'stick' at infinity. They may be read, and are cleared to 0 by the 'Read_and_Clear' operation. Statistics are time-averaged measures of some facet of system operation. They may be read. Values are read-only numbers which are not parameters, counters or statistics. The following are examples of several objects in various layers:

Examples of Data Link Objects:
- Number of packets transmitted
- Number of packets received
- Number of CRC errors

Examples of Network Objects:
- Number of networks
- List of network addresses

Examples of Transport Objects:
- Number of existing connections
- Retransmission time of any connection

Examples of Session Objects:
- Number of process names recognized
- Number of session virtual circuits open

Examples of Application Objects:
- Number of existing file connections
- Characteristics of various terminals in use

6.4.1 Layer Management Interfaces

The NMF has to access the internal data base of every other layer. Thus, each layer has to provide an interface in which the NMF can access its data base. NMF should be able to read any element of a layer's data base, set any element of a layer's data

base to a desired value and read and clear any element of a layer's data base. 'Read and Clear' is an indivisible operation provided to ensure that no race conditions exist. Each layer provides the three interfaces to the NMF described below.

Note that in the following definitions the term 'layer' refers to an entity within the layer. For instance, a node might have two data links. In this case, 'layer' would also specify which of the two data links are being referenced.

Layer_Read (Object, Modifier, Buffer_Pointer)

Function: The value of Object is placed at Buffer_Pointer

Where:

'Layer' is an entity within any one of the layers with which the NMF interfaces with - DLL, NL, TCL, SL, AL.

'Object' is the identifying number of a Network Management object.

'Modifier' is a number to be used in conjunction with Object to locate the particular value to read. The meaning of modifier can be different for different objects.

'Buffer_Pointer' is a pointer that points to the buffer in which to place the returned data.

Returns a value indicating one of the following:

Successfully completed,
Not done,
Bad parameter.

Layer_Read_and_Clear (Object, Modifier, Buffer_Pointer)

Function: The value of Object is placed at Buffer_Pointer and the value is then set to 0. This is to be considered as one indivisible operation.

Where:

'Layer' is an entity within any one of the layers with which the NMF interfaces - DLL, NL, TCL, SL, AL.

'Object' is the identifying number of a Network Management object.

'Modifier' is a number to be used in conjunction with Object to locate the particular value to read. The meaning of modifier can be different for different objects.

'Buffer_Pointer' is a pointer that points to the buffer in which to place the returned data.

Returns a value indicating one of the following:

Successfully Completed
Not Done
Bad Parameter

Layer_Set (Object, Modifier, Value_Pointer)

Function: Object is set to the value contained in Value_Pointer

Where:

'Layer' is an entity within any one of the layers with which the NMF interfaces - DLL, NL, TCL, SL, AL.

'Object' is the identifying number of a Network Management object.

'Modifier' is a number to be used in conjunction with Object to locate the particular value to read. The meaning of modifier can be different for different objects.

'Value_Pointer' is a pointer that points to the buffer which contains the value to be assigned to Object.

Returns a value indicating one of the following:

Successfully Completed

Not Done

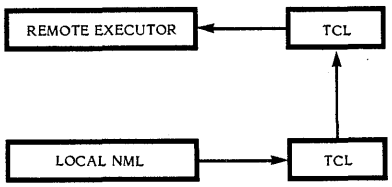
Bad Parameter

6.4.2 Other Interfaces

The Network Management Facility (NMF) uses the services provided by the other layers to communicate with the NMF residing on remote nodes. However, to the other layers, the NMF is just another user of its services and that layer does not have to provide the NMF with any special services. The NMF uses the standard functions which that layer provides.

6.4.3 Layer Management on Remote Nodes

To carry out an operation on a remote node, the NMF uses the services of the Transport (Control) Layer (TCL). It has to establish a connection with the NMF on the remote node, which (the remote node) will carry out the operation. The process on the remote node which carries out the command is termed the Remote Executor. The NMF first does an Active_Open. The remote transport address is the transport address of the Remote Executor (see illustration below). The Remote Executor always has an unspecified Passive_Open with a local TSAP id of 0003. A connection is established between the Active_Open on the local node and the Passive_Open on the Remote Executor. The NMF formats a command block, transmits it to the Remote Executor and waits for the response block from the Remote Executor.



The local NMF then transmits the command. The Remote Executor executes the command and returns a response. On receipt of the response, both of the NMFs close the virtual circuit. The formats of the command and response are shown below:

```
declare rb structure (
    TYPE        byte,
    NUMBER      byte,
    OBJECT      word,
    MODIFIER    word,          repeated NUMBER times
    LENGTH      word,
    VALUE       (LENGTH) byte );
```

COMMAND BLOCK FOR READ, SET, READ_AND_CLEAR BETWEEN LOCAL AND REMOTE NMFs

'TYPE' is the function code for this request:

0 - READ function

1 - READ_AND_CLEAR function

2 - SET function

'NUMBER' is the number of objects in this request.

The following fields are repeated for each of NUMBER objects:

'OBJECT' is the identifying number of a Network Management Object.

'MODIFIER' is a number to be used in conjunction with OBJECT to locate the particular value to return. The interpretation of MODIFIER can be different for different OBJECTS. The values that MODIFIER can take are listed along with the Network Management Objects in the document 'NMF Objects and Events'.

'LENGTH' is the length in bytes of the VALUE field. In the case of READ and READC commands, this field should contain a 0.

'VALUE' is the new value for OBJECT on a SET command. This field is ignored by NMF for READ and READ_AND_CLEAR commands.

The response transmitted by the remote NMF is similar to the above format. The format of the response is illustrated by the following PLM86 structure:

```
declare response_block structure (
    NUMBER          byte,
    OBJECT          word,
    MODIFIER        word,           repeated NUMBER times
    LENGTH          word,
    VALUE           (LENGTH) byte );
```

The fields in the response block illustrated above have the same meaning as those in the command block. If an object in the command block is illegal, it is not present in the response block. If the TYPE in the command block was READ AND CLEAR, and the object is not clearable, it is again not present in the response block. In the case of a SET command, the remote NMF attempts to set the value of the OBJECT to that specified in the VALUE field and then read the value of the OBJECT into the response block. Thus, if the object is not settable or it is not possible to set the OBJECT to the value specified in the value field, the VALUE in the response block will be different from that in the command block.

In a NMF implementation, the process that accepts commands from remote NMFs, the Remote Executor, should follow the following outline:

Do forever;

1. Issue unspecified Passive_Open. The local TSAP id is 0003H. The remote host id is unspecified.
2. Accept any connection.
3. Receive a message over the virtual circuit.
4. If message length is . 100 bytes, then go to step 8.
5. If error code is returned in step 3, then go to step 8.
6. Fill a buffer with the response. The size of this buffer is 200 bytes.
7. Transmit the buffer to the remote NMF.

8. Close the virtual circuit.

End do.

If there are two command blocks to be transmitted, NMF has to establish a virtual circuit twice since after processing the first command block, the remote NMF closes the virtual circuit. The overhead involved in doing this is tolerated since frequent transmission of command blocks is not anticipated.

The NMF process which transmits the commands follows the following algorithm:

1. Issue an Active_Open. The remote host id is that of the remote node and the remote TSAP id is 0003H.
2. Wait a finite time for the connection to get established.
3. If timeout occurs in step 2, then go to step 7.
4. Transmit the command block in the format described. The size of this block must be less than 100 bytes.
5. Wait a finite time for response buffer. The maximum size of this buffer is 200 bytes.
6. If a timeout occurs in step 5, then go to step 7.
7. Close the virtual circuit.

6.5 DOWN-LINE LOADING

The NMF provides the facility of down-line loading remote systems. This facility can be used for various reasons. Stations without local mass storage can use it to boot themselves. A station can force a remote station to boot (or re-boot) itself. This facility could be used so that a set of nodes boot from the same version of software. It could also be used to load a data base from (or to) a remote node.

A down-line loading operation requires the cooperation of two stations - the target node, the node which is to be loaded; and the executor node, the node which supplies the target node with the required data. A simple protocol is followed by the two stations. The target node is usually in a state where it can only use minimal data link facilities (it may be the communication system that is being loaded). The protocol used makes use of the raw data link facilities.

The process which runs on the target station is termed the Boot Consumer (BC) and the process on the executor station is called the Boot Server (BS). The two processes together provide a service which is general enough to be used for purposes other than booting; the names assigned to them are slightly misleading.

The boot consumer transmits requests to the boot server and then waits for the boot server to respond. The response from the boot server would typically consist of some data and some control information. The control information informs the boot consumer as to how the data is to be interpreted and whether more data is to follow. A typical boot sequence would consist of the boot consumer issuing a request for the first block of data, receiving a packet from the boot server, processing the packet and then issuing a request for the next block of data. This sequence of events continues until the loading operation is complete.

The protocol, followed by the two processes, can be broken down into three phases: Initiation, Connection Establishment and Software Loading.

6.5.1 The Initiation Phase

The initiation phase can be started by any node on the network, including the target node itself.

A remote node can start the initiation phase by transmitting a packet containing the 'Do Remote Load' command to the target station. Included in this command is the class code, a 16 bit value that indicates what exactly the target node has to load. The boot consumer on the target node responds with a 'Do Remote Load' response packet and then proceeds to the next phase. The formats of the 'Do Remote Load' command and response packets are illustrated below.

The BC on the target node can initiate the loading operation itself. It may be programmed to initiate the operation at system reset time, or at any other time that it desires.

Do Remote Load Command:

Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	5
DSAP ID	1 byte	=	NML SAP
SSAP ID	1 byte	=	NML SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	3H (Do remote load cmd)
Class Code	2 bytes	=	A 16 bit value that the target node is to use when requesting boot from the boot server
Padding			To meet minimum packet size

Do Remote Load Response:

Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	5
DSAP ID	1 byte	=	NML SAP
SSAP ID	1 byte	=	NML SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	01H (Do remote load resp)
Padding			To meet minimum packet size

6.5.2 The Connection Establishment Phase

The Connection Establishment phase consists of the BC issuing a Multicast bootstrap request message via the data link. The multicast addresses used by Intel products is 01 AA 00 FF FF FF. Included in the request message is the class code, denoting the type of data that the BC needs. The class code is a two byte field. For Intel systems, the most significant byte specifies the type of communication system while the least significant byte specifies the type of host.

If present, the Boot Server(s) checks to see if it can satisfy the request (it has the files required by the class code and the resources to satisfy the request). If so, it responds with a bootstrap acknowledge message. If it cannot satisfy the request or is not present, no acknowledge message is transmitted. If the Boot Server(s) does not respond within a fixed time period (one second), the BC reissues the request. It tries three times before aborting the process.

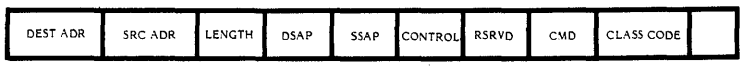
The boot consumer can get more than one boot acknowledge packet since there can be more than one boot server present on the network willing to boot the node. The boot consumer notes the source address of the first boot acknowledge packet and directs all further requests to this address.

The protocol observed by the boot consumer in the connection establishment phase is presented below:

1. Tries = 0
2. Tries = Tries + 1
3. If Tries = 4, then return (Error - no response received)
4. Transmit packet containing boot request and class code
5. Wait one second for a response packet
6. If timeout in step 5, then go to step 2
7. If not boot acknowledge packet, then go to step 2
8. Expected response has been received. Note the source id of the boot acknowledge packet. Ignore any boot acknowledge packets that may arrive later.

The format of the boot request and boot acknowledge packets is illustrated below:

Boot Request:



Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	7
DSAP ID	1 byte	=	NML SAP
SSAP ID	1 byte	=	NML SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	4H (Boot request)
Class Code	2 bytes	=	A 16 bit value that contains the code of the requesting device
Padding			To meet minimum packet size

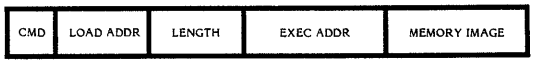
Boot Response:

Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	5
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	5H (Boot response)
Padding			To meet minimum packet size

6.5.3 The Loading Phase

When a connection has been established between the boot consumer process and the boot server process, the actual loading operation begins.

The protocol allows any number of modules to be loaded. The format of a module is shown below:



Command	1 byte: Indicates if the module is to be executed and if there are more modules to be loaded. Bit 0=1 indicates there is another module to be loaded. Bit 1=1 indicates that the execution address is to be called.
Load Address	4 bytes: Specifies the first address where the data is to be placed.
Length	2 bytes: Specifies the length of the memory image data.
Execution Address	4 bytes: Specifies the execution address of the loaded memory image.
Memory Image	The memory image to be loaded. It can be between 0 and 2**16 - 1 bytes long.

From the class code, the boot server knows exactly which modules the boot server needs. The boot server builds up a file of the following format:

```
< Header 2 >< Module 1 >  < Header 2 >< Module 2 > . . .      < Header n >< Module n >
| Block 0 |      Block 1 |                                          | Block m |
```

The boot server breaks the file up into an arbitrary number of blocks of arbitrary size. The size of each block should, however, be small enough to fit in a boot response packet (illustrated in section 6.9.5). Upon receipt of a boot request for any block x, the boot server merely transmits block x to the boot consumer.

The boot consumer transmits requests for blocks of data. It waits one second for the boot server to transmit the data. If no response is received from the boot server, the BC retransmits the request. It makes three attempts before aborting the process. Upon receipt of a response from the BS, it processes the response and then issues a request for the next block of data.

The header of each module informs the BC of the length of the memory image of the module and the location where the module is to be placed. If the module header indicates that the module is to be executed, then the execution address is called as a subroutine. If the module indicates that another module is to follow, then after the current module has been received, the BC continues with its protocol to obtain the next module.

The detailed algorithm that the BC follows is presented below. To increase the readability of the algorithm, it has been broken up into three subroutines.

The variables used in the three subroutines are:

Block_number	- The expected block number of data from the boot server.
Buf_data_length	- The length of the data in the receive buffer.
Buf_address	- The address of the receive buffer.
Last_buffer	- If true, then the present buffer is the last one for the current module.
Status	- Can take three values: OK; No response from boot server; and End of file.

```
Loading_phase: procedure;

/*
    This model uses two buffers, one to store the module header and another to
    store received packets from the boot server.
*/
    Block number = 0; /* Length of data in receive buffer */
    Buf_Data_Length = 0; /* Length of data in receive buffer */
```

```
L0: Last_buffer = false;

    Status = Read_info (11 bytes, into module_header_buf);
    /* Read the header fields of a module */

    If Status , . OK then return (Error - Status);

    Status = Read_info (mod header_buf.length, mod_header_buf.load_addr);
    /* Read the memory image starting at location specified in mod header */

    If status  , . OK then return (Error - Status);

    If mod_header_buf.CMD = . Execute then call mod_header_buf.exec_addr;
    /* if the module is to be immediately executed, then execute it */

    If mod_header_buf.cmd = . More, then go to L0;
    /* If another module is to follow, then continue with the protocol */

    Return (OK - Loading complete)

    End loading_phase;

Read-info:procedure (Number_bytes, Starting_loc);
/*
    Receives and stores 'number_bytes' bytes from boot server and stores them
    starting at location 'starting_loc'.
*/

Do while number_of bytes , . 0;
/* there are more bytes to be stored */

    if BUF_data_len = 0, then
    /* Buffer is empty, get next block of data from boot server*/
```

```
            if (Status := Get_buffer) , . OK then return (Status);
            /*
            Buf_data_len bytes have been received and stored at Buf_address
            */

        Bytes_to_move = min (Number_bytes, Buf_data_len);
        call move (starting_loc, Buf_address, bytes_to_move);
        Number_bytes = Number_bytes - Bytes_to_move;
        Buf_data_len = Buf_ data_len - Bytes  to_move;
        Starting_loc = Starting_loc + Bytes_to_move;
        Buf_address = Buf_address + Bytes  to_move;

end while:

return (OK - 'Number_bytes' read and stored at 'Starting_loc');
end Read_info;
Get_buffer:procedure;
/*
Receives a packet from the BS for the next block of data
*/
If Last_buffer, then return (End_of_file);
L0:  Tries = 0;
        Tries - Tries + 1;
        If Tries = 4, then return (Error - No response from boot server);
        Transmit boot request packet to boot server for 'Block_number' block packet
        Wait for 1 second;
        If timeout, then go to L0;
        If received packet not for 'Block  number' block, then go to L0;
/*
Packet for block 'Block  number' has been received from boot server
*/
If received packet contains boot end of data response
then Last_buffer = True;
Buf_address = Address of first byte of data in received packet;
Buf_data_len = length of data in the received packet;
```

```
Block number = Block_number + 1; /* The next expected block number */
return (OK);
end Get_buffer
```

Note that modules that are executed immediately can access data from the boot server by using the procedure 'Read_info'. The parameters for this procedure are the starting address of the location at which the received data is to be placed and the number of bytes of data that are required. If the latter parameter is set to infinity, then READ info continues to place received data from the boot server until it receives a packet from the boot server containing the 'end of data' command.

The above feature allows modules to get at data in a different format than the boot module format described earlier. Using this feature, it is possible to first load a Loader. The Loader begins execution and uses the procedure 'Read_info' to receive data coming over the net and interprets the data in any way that it likes.

The formats of the various packets used in the loading phase are presented below:

Boot Data Request:

Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	7
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	6H (Boot data request)
Block	2 bytes	=	The block number of the boot file that is requested.
Padding			To meet minimum packet size

Boot Data Response:

Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	Length of data field + 7
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	7H (Boot data response)
Block	2 bytes	=	The block number of the boot file that is contained in the data field.
Data	n Bytes	=	The command fields and data to be sent
Padding			To meet minimum packet size

Boot End-of-Data Response:

Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	Length of data field + 7
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	2H (Boot end of data resp.)
Block	2 bytes	=	The block number of the boot file that is contained in the data field

Data	n Bytes	=	The command fields and data to be sent
Padding			To meet minimum packet size

6.6 ECHO SERVICE

The echo service is used to determine the presence of a node on a network, the communication path to the remote node, to ascertain the viability and functionality of the remote host and to determine the class code of the remote station.

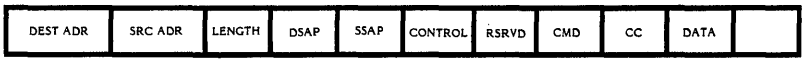
On receipt of the echo command, the NMF generates an echo response packet. The formats of these packets are illustrated below. The NMF copies the data field from the command packet to the data field of the response packet. It is up to NMF to copy as much of the data from the command packet into the response packet as possible. Also included in the response packet is the class code which identifies the type of node and the software that is running.

The echo response packet should be generated immediately since the remote NMF only waits a small finite time for it.

Echo Command:

Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	7
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	8H (Echo Command)
Data	x bytes	=	x bytes of data
Padding			To meet minimum packet size

Echo Response:



Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	9
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	9H (Echo response)
Class Code	2 bytes	=	The code that identifies the type of station and the software running on it
Data	y Bytes	=	The same data as in the command (y , =x)
Padding			To meet mimimum packet size

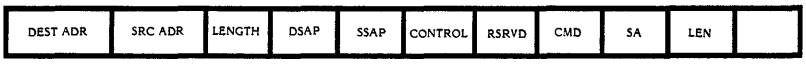
6.7 UP-LINE DUMPING AND RESET

The NMF provides the up-line dumping facility to enable a node to get a dump of the memory of a remote node.

On receipt of the dump command, the NMF generates a dump response packet. The formats of these two packets are illustrated below. NMF provides a dump of the memory beginning at the starting address specified in the dump command packet. If the dump length specified is too big (the memory image would not fit in a maximum sized packet), the NMF copies only that much of the memory that would fit in a packet. A remote node would have to transmit a series of dump commands to get a dump of a large area of memory.

The dump response packet should be generated immediately since the remote NMF only waits a small finite time for it.

Dump Command:



Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	0BH
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	AH (Dump Command)
Start Address	4 bytes	=	The starting address of the memory to be dumped
Len	2 bytes	=	The length of the memory area in bytes to be dumped
Padding			To meet mimimum packet size

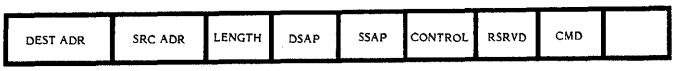
Dump Response:

Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	0BH + Len
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	BH (dump response)

Start Address	4 bytes	=	The start address of the memory area dumped
Len	2 bytes	=	The length of the memory area dumped
Memory Image			The memory image
Padding			To meet mimimum packet size

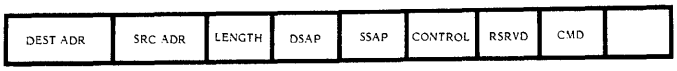
Another feature that the NMF provides is the reset service. On receipt of a packet containing the reset command, the NMF generates a reset acknowledge packet and then resets the processor. This feature is included in this section because it is anticipated that it will be used in conjunction with the up-line dumping facility. It may be possible to reset a remote system that is behaving inappropriately by issuing a reset command and then examining the memory of the remote system using the dump commands mentioned earlier. The formats of the packets are shown below.

Remote Reset Command:



Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	5
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	0FH (Do remote reset cmd)
Padding			To meet mimimum packet size

Remote Reset Response:



Destination Address	6 bytes		
Source Address	6 bytes		
LLC PDU Length	2 bytes	=	5
DSAP ID	1 byte	=	NMF SAP
SSAP ID	1 byte	=	NMF SAP
Control	1 byte	=	0C3H (MUI)
Reserved	1 byte	=	0
Command	1 byte	=	0EH (Do remote reset resp)
Padding			To meet mimimum packet size

6.8 READ/SET MEMORY COMMANDS (ON LOCAL AND REMOTE NODES)

The NMF provides the host with the ability to read/set memory of any host present on the network. The service provided is:

Read_Memory (Node, Memory_Pointer, Length, Buffer_Pointer)

Where:

Node	- Is the node on which this operation is to be performed.
Memory_Pointer	- The starting location of the memory to be read.
Length	- The length in bytes of the memory to read.
Buffer_Pointer	- The buffer in which to store the values read.

Returns a value indicating one of the following:

Successfully Completed
Not Done
Bad Parameter

Set_Memory (Node, Memory_Pointer, Length, Value_Pointer)

Where:

Node	- Is the node on which this operation is to be performed.
Memory_Pointer	- The starting location of the memory to be set.
Length	- The length in bytes of the memory to set.
Value_Pointer	- Pointer to buffer which contains the values to be written at Memory_Pointer.

Returns a value indicating one of the following:

Successfully Completed
Not Done
Bad Parameter

READ_MEMORY/SET_MEMORY commands on remote nodes are carried out in a manner similar to that for the READ, READ_AND_CLEAR and SET commands described earlier. The NMF utilizes the services of the Transport Control Layer to communicate with the NMF on the remote node. The only difference is in the command and response blocks which are illustrated below:

```
declare rb structure (
    TYPE            byte,
    LENGTH          word,
    START_ADDRESS   pointer,
    VALUE           (LENGTH) byte);
```

COMMAND BLOCK FOR READ/SET MEMORY
BETWEEN LOCAL AND REMOTE NMFs

TYPE - The function code for this request.

3 - READ_MEMORY function
4 - SET_MEMORY function

START ADDRESS - Pointer that points to the first address where the function is to be performed.

LENGTH - The length in bytes of the memory to be read/written.

VALUE - Valid only for the SET MEMORY command. It contains the values to which the memory is to be set.

In the case of the READ MEMORY function, the response block contains the memory that is to be read. In the case of SET MEMORY, there is no response block.

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.