

**SYNCHRONOUS
POINT-TO-POINT
PROTOCOL**

XEROX

Notice

This *Xerox Network Systems Standard* describes the Synchronous Point-to-Point Protocol.

This document is being provided for informational purposes only. Xerox makes no warranties or representations of any kind relative to this document or its use, including the implied warranties of merchantability and fitness for a particular purpose. Xerox does not assume any responsibility or liability for any errors or inaccuracies that may be contained in the document, or that the use of the information herein will function or perform in an intended manner.

The information contained herein is subject to change without any obligation of notice on the part of Xerox.

All text and graphics prepared on the Xerox 8010 Information System.

Copyright © 1984, Xerox Corporation. All rights reserved.

XEROX®, XNS, Interpress, Clearinghouse, and all XEROX products mentioned herein are trademarks of XEROX CORPORATION.

Printed in U.S.A.

This document is one of the family of publications that describe the network protocols underlying Xerox Network Systems (XNS).

Xerox Network Systems comprise a variety of digital processors interconnected by means of a variety of transmission media. System elements communicate both to transmit information between users and to economically share resources. For system elements to communicate with one another, certain standard protocols must be observed.

Comments and suggestions on this document and its use are encouraged. Please address communications to:

Xerox Corporation
Xerox Network Systems Institute
2300 Geng Road
Palo Alto, California 94303

1. Introduction	1
1.1 Purpose	1
1.2 Applicability and relationship to other protocols	2
1.3 Notation	2
1.4 Document organization	3
2. Overview	5
2.1 Nature of Level 0 protocols	5
2.2 Nature of the Synchronous Point-to-Point Protocol	6
2.3 The communication facilities	6
2.4 Communicating entities	7
2.4.1 System elements	7
2.4.2 Interfacing system elements	8
2.4.3 Associations	8
3. Protocol description	9
3.1 Description of Level 0	9
3.1.1 Physical layer	9
3.1.2 Data link layer	10
3.2 Framing	10
3.2.1 Flag sequence	10
3.2.2 Packet field	10
3.2.3 Frame check sequence	11
3.2.4 Transparency	11
3.2.5 Frame transmission	11
3.3 Packet formats	12
3.3.1 Control packets	12
3.3.2 The Data packet	18
3.3.3 Private packets	19
3.4 Half-duplex medium contention	19
3.5 service interface	20

4. The synchronous automaton	23
4.1 The states of the automaton	23
4.1.1 Idle phase	23
4.1.2 Option negotiation phase	24
4.1.3 Data transfer phase	24
4.1.4 Termination phase	25
4.1.5 Loss of synchronization	25
4.2 Events	25
4.2.1 Client operations	26
4.2.2 Packet arrivals	26
4.2.3 Medium status indications	26
4.2.4 Timeouts	27
4.3 State transitions	27
Appendices:	
A. References	31
B. Type assignment procedures	32
C. Assigned packet types	33

LIST OF FIGURES AND TABLES

Figures:

1.1	Packet descriptions	3
2.1	Level 0 of Network Systems	6
2.2	Examples of internetworking configurations	8
3.1	ISO-OSI reference model of Levels 0 and 1	9
3.2	Level 0 frame format	11
3.3	The Control packet format	12
3.4	The MyOptions packet format	14
3.5	The OptionsAck packet format	14
3.6	The VersionReject packet format	15
3.7	The AddressReject packet format	15
3.8	The ClassReject packet format	16
3.9	The SizeReject packet format	16
3.10	The Null packet format	17
3.11	The AreYouThere packet format	17
3.12	The IAmHere packet format	17
3.13	The Terminate packet formats	17
3.14	The Data packet format	18
3.15	The Private packet format	19
4.1	Activating states	28
4.2	Deactivating states	29

Tables:

3.1	Control types	13
3.2	Allowed associations between entities	16
4.1	The Synchronous Point-to-Point State Table	30
C.1	Packet types	33

An important aspect of the Network Systems (NS) communication architecture is the interconnection of Ethernets and remote system elements to form an internetwork. A homogeneous internetwork allows resource sharing to occur among all the system elements attached to it. However, the internetwork is not homogeneous at the physical level; rather, there is a mixture of Ethernets and other interconnection media. A Level 0 protocol, like the one specified in this document, provides a datagram transmission service for a particular class of medium. Once datagrams can be transmitted throughout the internetwork, higher-level protocols can be employed as necessary for efficient and reliable use of the shared resources.

1.1 Purpose

This document describes the protocol for the interconnection of individual communicating entities or data terminal equipment (DTEs) over a data link. The data link is comprised of data communication equipment (DCEs) and the communication channel. The communication channel may be any of a variety of transmission facilities, such as private dedicated connections and public common-carrier connections. This protocol assumes that *bit-synchronous* transmission is used between DTEs. Bit-synchronous transmission relies on clocking, generated either in the DTEs or DCEs, to aid in the recognition of binary digits. The interface protocol between DTE and DCE may be any of a variety of common interface standards, such as those defined by the EIA RS-232-C, EIA RS-449, and CCITT V.35 standards.

This document is intended for implementors of the Synchronous Point-to-Point Protocol. It provides the details necessary to transmit NS Internetwork Datagrams (or any other Level 1 packet) between two communicating entities. It describes the allowable frame formats and protocol commands. It does not, however, describe any particular DTE-to-DCE interface standard. The reader will benefit from familiarity with the NS protocol hierarchy and specifically the NS Internetwork Datagram Protocol, as specified in *Internet Transport Protocols* [6].

The Synchronous Point-to-Point Protocol has been designed to meet the following goals:

- Simplicity. The protocol defines a simple connection between communicating entities, and is easily implemented.
- Transparency. In keeping with the concept of a layered architecture the protocol makes no assumptions about data that is transmitted, nor does it constrain the data in any way.
- Compatibility and maintainability. The protocol provides a version negotiation feature which allows communicating entities to agree on the specific version of the protocol to be used. This allows revisions to this protocol to be easily introduced.

- Error detection. The protocol detects transmission errors but makes no attempt to correct them. It relies on higher-level protocols to supply error correction when necessary.
- Duplexity. Both half-duplex and duplex communication modes are supported.
- Symmetry. For duplex mode the protocol is symmetrical, requiring no master/slave relationship. A master is defined for half-duplex mode.
- Switched or non-switched media. The protocol functions equally well over switched and dedicated types of media. Establishing the switched physical connection is beyond the scope of this specification.

1.2 Applicability and relationship to other protocols

The Synchronous Point-to-Point Protocol represents one of any number of transmission media protocols in the NS protocol hierarchy. It corresponds to and serves the same function as the Ethernet Protocol [1], HDLC, and the X.25 protocol, to name others. Protocols at this level serve client protocols at the network (internetwork) level, NS Level 1. For a discussion of how the Synchronous Point-to-Point Protocol relates to other data link protocols see section 2.2. This standard is not intended to supplant HDLC in any application where that protocol is otherwise appropriate.

As a Xerox Standard, the Synchronous Point-to-Point Protocol is applicable to Xerox Forwarding System Elements, Cluster System Elements, Interfacing System Elements, and Terminal System Elements (see section 2.4 for definitions), when such system elements are equipped with bit-synchronous ports. This standard does not apply to asynchronous ports, byte-synchronous ports, or multi-dropped bit-synchronous ports.

1.3 Notation

Throughout this document, special fonts are used to depict certain Synchronous Protocol constructs instead of using quote marks or other delimiters. This convention aids the eye in discriminating between Synchronous Protocol text and the exposition. Items in **this font** indicate elements of the service interface of the Synchronous Protocol. The service interface makes the functionality of a protocol implementation available to its clients, higher-level protocol, and system management implementations.

Items in this font indicate packets defined by the Synchronous Protocol.

Service interface elements are referred to as operations; the notation used represents them as procedure calls. The definition of an operation follows a simplified Courier [5] syntax for defining procedure calls. Since the service interface must be general enough for varied implementations, only vague typing information is provided. The following demonstrates the syntax for the documentation of operations:

**SomeOperation: TYPE = PROCEDURE [argument1: Arg1Type, argument2: Arg2Type, ...]
RETURNS [result1: Result1Type, result2: Result2Type, ...]**

REPORTS [error1, error2, ...];

And the activate operation would be denoted as

Activate: TYPE = PROCEDURE [role: ActivateRole]

RETURNS []

REPORTS [alreadyActive];

ActivateRole: TYPE = {active, passive};

where the argument is **role**. The nature of **role** and the behavior of **Activate** in both the normal and abnormal cases would be described in the text.

Packets are described pictorially (Fig. 1.1) in terms of the *octet*, a sequence of eight binary digits. Each octet is diagrammed with the most significant bit on the left, the least significant bit on the right. Octets are displayed such that the top-to-bottom order represents the order in which they are transmitted. The order of bit transmission is dependent on the field of the packet, as described later.

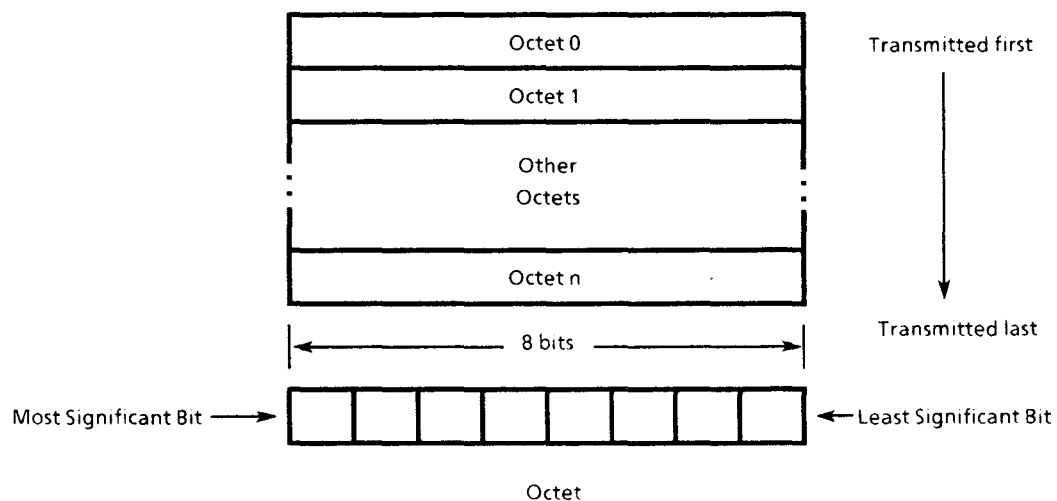


Figure 1.1 Packet descriptions

1.4 Document organization

Chapter 2 of this document gives an overview of the concepts defined in the standard. Chapter 3 defines the entire Synchronous Point-to-Point Protocol. Chapter 4 provides a description of the protocol in terms of a state diagram and a state transition table.

Appendix A lists other documents which supplement the specification. Appendix B describes the administrative procedures for obtaining ranges of packet types. Appendix C lists the currently assigned packet types.

The primary function of an NS Level 0 protocol is to move datagrams across a transmission medium. This level includes the lowest-level transmission medium-specific protocols, such as Ethernet. Level 0 may also include protocols used to transport Level 1 packets across a communication facility that is more independent of a class of medium, such as X.25 networks.

2.1 Nature of Level 0 protocols

At Level 0, the Level 1 packet is treated as data; the Level 1 packet is interpreted only by the next higher level. Level 1 packets are encapsulated for transmission in each packet, block, or frame of Level 0 format as per the conventions of the transmission medium protocol. Upon reception the transmission medium protocol specifies the inverse process of decapsulation, which preserves the structure of the Level 1 packet. The process of encapsulation and decapsulation must preserve this logical structure.

Packet encapsulation consists of transforming a Level 1 packet in whatever fashion is necessary to permit the entire packet to pass through the transmission medium transparently as "data." This generally involves simply adding transmission medium-dependent headers and trailers for addressing, error detection or correction, etc., but could also include transmission medium-specific fragmentation, retransmission, and encoding of data in various ways such as bit compression. The process of encapsulation is specific to a pair of Level 1 and Level 0 protocols.

In order to physically transmit a Level 1 packet over a transmission facility, the recipient must be addressed according to the conventions of the facility. When addressing differs from the Xerox NS addressing structure, a facility-specific address must be derived from the NS 48-bit host address. Also, the broadcast and multi-cast modes of addressing defined in the NS addressing structure must be supported via the facility. This may require special features in the Level 0 protocol to augment the features of the transmission facility.

When a Level 1 packet is received over a transmission medium, it is first decapsulated by applying the inverse of the encapsulation transformation. In some cases, this may involve packet fragment reassembly, code conversion, padding removal, etc. If the Level 1 packet is routed to another network, it is re-encapsulated and subsequently transmitted according to the conventions of this new transmission medium.

Level 0 protocols may be based on datagrams or virtual circuits. Virtual circuit protocols provide reliability at the data link level. They are common in network architectures that do not provide end-to-end reliability at higher levels. Datagram protocols rely on the reliability of higher-level protocols. NS Level 0 protocols are normally datagram oriented, due to the reliable end-to-end protocols, such as the Sequenced Packet Protocol (SPP) [6], at Level 2. These protocols are robust to the error rates that are characteristic of today's transmission facilities. Virtual circuit protocols at Level 0, such as X.25 and HDLC, offer redundant

reliability when used with NS SPP, and they incur performance costs that are not always justifiable. For a point-to-point data link that exhibits normal error characteristics, a datagram protocol is preferred.

2.2 Nature of the Synchronous Point-to-Point Protocol

The Synchronous Point-to-Point Protocol defines a mechanism for the "best effort delivery" of data packets over synchronous data links. It serves to interconnect various types of communicating entities, especially those for which the geographical distance or the entity's characteristics do not permit a direct connection via local area networks such as the Ethernet.

The protocol does not attempt to provide a high degree of reliability by correcting errors; rather, it merely discards received packets in which errors have been detected. It assumes that higher level protocols handle reliability, sequencing, and suppression of duplicates.

The protocol is based on encoding, framing, and error detection conventions that are supported by commercially available integrated circuits. These conventions are like those used in other bit-synchronous data link protocols, such as SDLC and HDLC. This allows inexpensive yet high-performance implementation of the protocol on a wide variety of communicating entities.

The NS architecture allows many different types of Level 0 protocols, reflecting various networking needs. Figure 2.1 places the Synchronous Point-to-Point Protocol among the other Level 0 protocols in the NS architecture. Like the Ethernet, it is a connectionless protocol that strives for peak performance from the normally errorless medium. When transmission medium errors do occur, higher-level protocols compensate to achieve reliability. In contrast, if full HDLC were used as the point-to-point data link protocol, both HDLC and the higher-level protocols would be incurring the overhead necessary to ensure reliability.

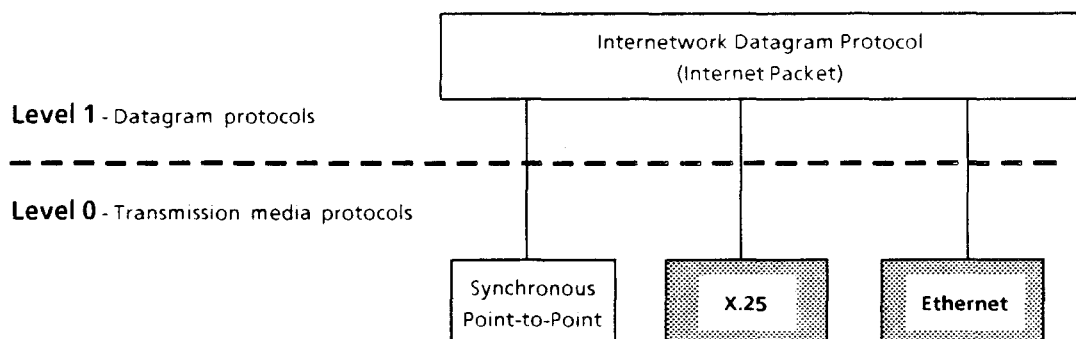


Figure 2.1 Level 0 of Network Systems

The relationship of the Synchronous Point-to-Point Protocol to other Level 0 protocols

2.3 The communication facilities

The communication facility used for the interconnection of individual communicating entities is one that allows the transmission of packets in bit-synchronous mode. The facility includes the DCEs and the medium. Various interface standards apply to the way in which DTEs link with DCEs, and how DCEs interface to media. Those standards are methods by

which the communication facilities are used, but are not of direct concern in this standard. This standard assumes that the communication facility provides a DTE-to-DCE convention that allows the following to be detectable by the DTE: a DCE ready indication, a remote DTE connection request for switched connections, and a clock signal for bit serialization and bit recognition. It is assumed that the DTE can indicate to the DCE that a connection can be established and that a connection should be broken. Communication facilities that satisfy these criteria are offered by common carriers, as switched circuits and leased circuits, and in many different forms as private media.

This protocol is suitable for both dedicated and switched circuits. Thus, the circuit on which the link depends may be permanent or temporary. The circuit must be point-to-point; it cannot be multipoint. Methods for establishing a circuit are outside the scope of this standard.

Both duplex and half-duplex modes of communication may be employed. For duplex mode, the protocol is symmetrical, requiring no master-slave relationship; for half-duplex mode, a master is defined. The protocol provides a method for determining the master-slave relationship dynamically. Both duplex and half-duplex are supported for both dedicated and switched circuits.

2.4 Communicating entities

A *communicating entity* (CE) is a station or system element that is capable of communicating with another. This includes systems elements that are attached to the Ethernet and stand-alone stations that use telephone lines to connect to other stations. Communicating entities fall into two classes, system elements and interfacing system elements. Possible configurations of these different entities are shown in Fig. 2.2.

2.4.1 System elements

System elements are hosts that have been assigned 48-bit IDs or *host addresses*. This address is the physical address of a station as defined in [6]. System elements are capable of implementing the full range of NS protocols, and can be both the source and destination of packets. System elements may act as terminal system elements or forwarding system elements. *Forwarding system elements* use a Level 0 protocol for interconnecting Ethernets, in which case they are referred to as *internetwork routers*. A *cluster system element* is a forwarding system element that allows multiple terminal system elements to link to it via a Level 0 protocol, forming a *cluster network*. *Terminal system elements* are not directly connected to the Ethernet, and use a Level 0 protocol to link to it via either cluster system elements or interfacing system elements. (Terminal system elements may be connected to the Ethernet, but do not or cannot make use of that connection to communicate with other terminal system elements.) Terminal system elements are occasionally referred to as *remote system elements* (in which case they are not connected to the Ethernet).

The NS architecture allows participation by any system element that implements the standard NS protocols and functions. One of the assumptions made about system elements is that they possess unique host addresses. There are, however, "anonymous" hosts which do not have unique 48-bit host addresses assigned to them, but which are otherwise capable of implementing the NS communication architecture. The Level 0 protocol facilitates the physical interfacing of these hosts by allowing them to acquire an address when they initiate communication with another communicating entity.

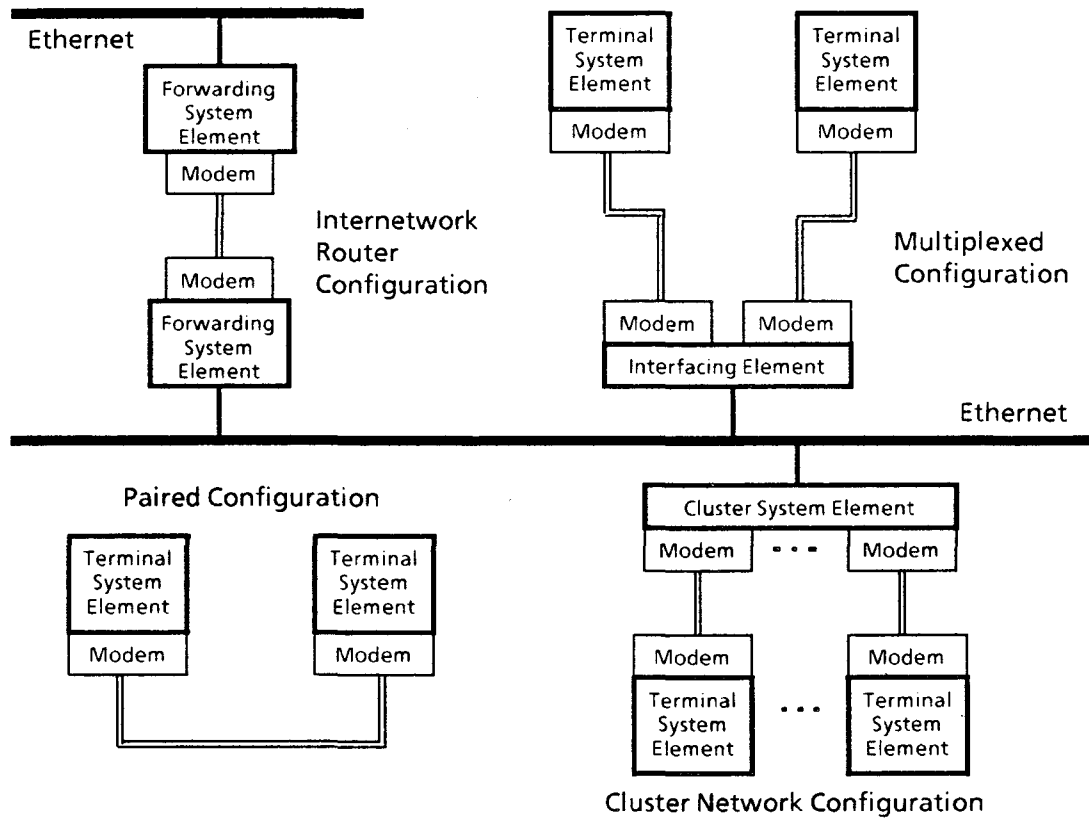


Figure 2.2 Examples of internetworking configurations

The double lines indicate media where the Synchronous Point-to-Point Protocol would be used.

2.4.2 Interfacing system elements

Interfacing system elements have no host address and thus are never the source nor the addressed destination of packets. They switch packets from one medium to another, translating Level 0 protocols with no knowledge of the higher level protocols. The interfacing system elements rely on addressing information provided in the Level 0 packets to perform the switching function.

2.4.3 Associations

The connection established between two compatible communicating entities is called an *association*. Not every kind of communicating entity can establish an association with a given communicating entity. The allowable associations are enumerated in section 3.3.1.4.

The Synchronous Point-to-Point Protocol has many components: packets, frames, and physical and service interfaces. The definition of each of these is set forth below. The next chapter describes, by means of a finite state automaton, how these components come together.

This document describes version 3 of the Synchronous Point-to-Point Protocol.

3.1 Description of Level 0

As suggested by the International Standards Organization's Open Systems Interconnection (ISO-OSI) reference model, it is useful to consider Level 0 as being made up of two layers, the *physical* layer and the *data link* layer. The Level 1 protocols, which rely on this protocol, will be referred to collectively as the *client* layer. Figure 3.1 illustrates the relationship of the different protocol levels.

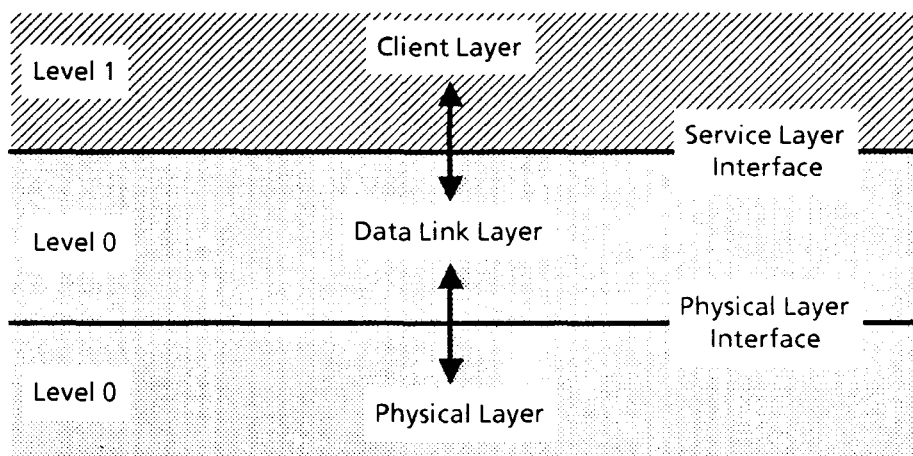


Figure 3.1 ISO-OSI reference model of Levels 0 and 1

The diagram shows the relationship between the levels and layers in the reference model.

3.1.1 Physical layer

The Physical Layer Interface may be any standard DTE-to-DCE interface that provides a dedicated or switched circuit for bit-synchronous transmission between DTEs. Services assumed from the Physical Layer include passive operations to detect that the DCE is ready and that a circuit connection request from a remote DTE has arrived. Active operations include the ability to indicate that the DTE is ready for a connection and that the DTE wishes to terminate a connection.

The Synchronous Point-to-Point Protocol does not impose restrictions on the transmission rate. Data transmission rates are limited only by the medium and the DTE-to-DCE standard used to interface to it.

Although the Synchronous Point-to-Point Protocol assumes that packets may be damaged in transmission, excessive error rates at the Physical Layer may reduce performance of higher protocol levels.

3.1.2 Data link layer

The Synchronous Point-to-Point Protocol transports Level 1 protocol packets by encapsulating them according to a bit-synchronous framing format [4]. Detection of transmission errors is achieved by computing a frame check sequence (FCS) as in HDLC. The protocol does not provide end-to-end error notification; damaged packets are simply discarded.

The protocol may be used in both duplex and half-duplex mode. Half-duplex contention is resolved immediately after the physical connection is established. A mechanism is provided for alternating the right to send (see section 3.4).

The protocol supports the creation of associations between communicating entities by allowing address exchange and parameter negotiation. Termination of associations may be initiated by either partner.

The NS architecture requires that a 48-bit host addressing scheme be used. In cases where layers above Level 0 use other addressing schemes, they are responsible for translating their addresses into the required unique 48-bit addresses before handing their packets to the Level 0 implementation.

3.2 Framing

The framing format defined by the protocol is the *bit-synchronous framing format*. This format defines how packets are delimited, how they are transmitted transparently, how they are error-checked, and how inter-frame gaps are encoded. The format for a frame is shown in Fig. 3.2.

3.2.1 Flag sequence

All frames start and end with a *flag sequence* field of one zero bit followed by six contiguous one bits and one zero bit; that is, it has the value 01111110_2 .

3.2.2 Packet field

The *packet* field contains an integral number of up to (but excluding) $65,536 (= 2^{16})$ octets, limited by a negotiated maximum packet size. The protocol defines several different packet formats for the packet field (see section 3.3). The formats fall into three general categories: control packets, data packets, and private packets. Control packets are used for option

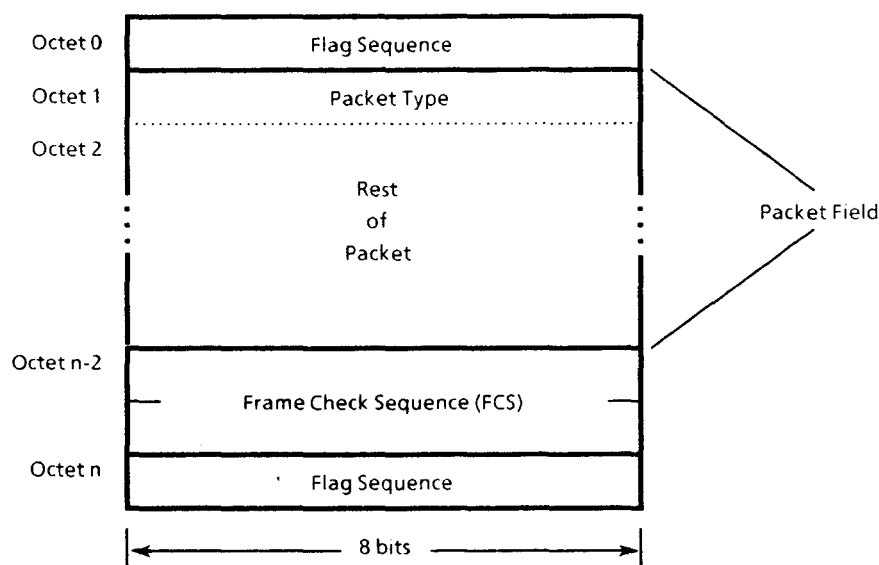


Figure 3.2 Level 0 frame format

The packet type field is defined but not interpreted at this level to insure uniform placement within the packet and uniform value assignment.

negotiation, half-duplex line turn-around in the absence of data flow, and termination handshakes. Data packets carry Level 1 packets. Private packets allow for private formats and semantics. Their specification is outside the scope of this document.

3.2.3 Frame check sequence

The standard HDLC *frame check sequence* field (FCS) follows the packet field. It is used to detect transmission errors. The check sequence is applied to all of the bits of the packet field. The algorithm for computing the FCS, and implementation notes are given in section 3.6 of [4].

3.2.4 Transparency

All octets of the packet field and FCS are transmitted using the following transparency rules: the transmitter must insert a zero bit after all sequences of five contiguous one bits (including the last five bits of the FCS) to ensure that a flag sequence is not inadvertently inserted. The receiver must discard any zero bit which directly follows five contiguous one bits. The inserted bits are not included in the FCS computation.

3.2.5 Frame transmission

The octets of a frame are transmitted in the order of most significant octet first, least significant last. Except for the FCS, each octet is transmitted least significant bit first. The FCS is transmitted most significant bit first.

A frame having more than the agreed upon maximum size is considered an invalid frame and should be discarded by the receiver. A frame can be aborted by transmitting at least seven contiguous one bits (with no inserted zeros).

Interframe gaps can be filled with either flags or an idle condition (all binary 1s). It is slightly preferable to use line idling when using a half-duplex communication facility. Since half-duplex DCEs usually remove transmission carrier when not transmitting, the stream of synchronous bits is abruptly truncated. The partially damaged bits that result will appear as aborts, followed by loss of carrier, when line idling is used for interframe fill. If flags are used for interframe fill with half-duplex DCEs, the damaged fragments may instead appear as a beginning of frame, followed by a byte of random data, followed by loss of carrier. This may cause some implementations to incur more overhead during line turn-around.

3.3 Packet formats

As mentioned previously, there are three classes of packets. They are distinguished by the first octet of the packet field. The definitions of these packet types and the format of the associated packets are given below.

3.3.1 Control packets

Control packets have a common structure for the first two octets, shown in Fig. 3.3. A Control packet is indicated by a packet type of 0. Because Control packets are the first packets exchanged on a connection in the Synchronous Protocol, and because the first octet is zero, the format of this packet, and consequently the protocol of this connection, can be distinguished from other synchronous protocols such as SDLC [3] and HDLC [4].

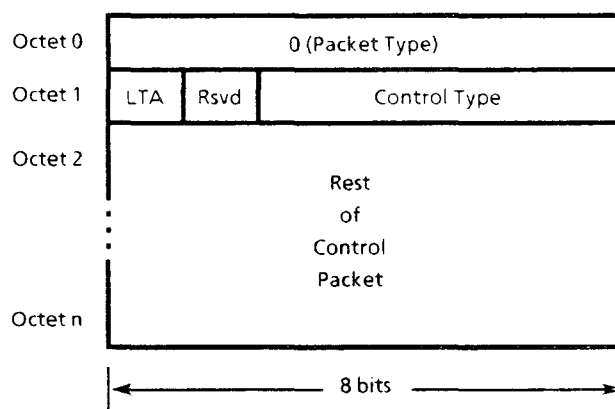


Figure 3.3 The Control packet format

The first bit of the second octet of the packet is the *line turn-around* (LTA) indicator. For duplex media, the indicator is ignored. For half-duplex media the bit is 0 until the right to send is yielded to the receiver, at which time it becomes 1. See section 3.4 for a discussion of the use of this field with half-duplex media. The second bit (Rsvd) is reserved; its value is not specified and should be ignored. The remainder of the octet holds the *control type*, the decimal values of which are shown in table 3.1.

Table 3.1 Control types

Control Type Name	Control Type (decimal)
AddressReject	14
AreYouThere	4
ClassReject	13
IAmHere	5
MyOptions	10
Null	1
OptionsAck	11
SizeReject	15
TerminateReply	3
TerminateRequest	2
VersionReject	12

The control type determines the structure of the control packet beyond the first two octets. These different types of control packets are referred to by their control type names. The types ending in "Reject" are referred to collectively as Reject control packets, the types beginning with "Terminate," as Terminate control packets.

3.3.1.1 The MyOptions control packet

The protocol defines an option negotiation phase for protocol version number exchange, host address assignment for anonymous hosts, host classing, and maximum packet sizing. The MyOptions control packet conveys these parameters. The format of the MyOptions packet is shown in Fig. 3.4.

The MyOptions control packet is usually the first packet sent by each communicating entity on a previously inactive or idle connection. The exchange is symmetrical; each entity must send at least one MyOptions packet. Each entity acknowledges the acceptance of the other's MyOptions by sending an OptionsAck (Fig. 3.5), or by sending one of the rejection packets. The exchange is asynchronous; after an entity has received an acceptable MyOptions packet, it may first send either its own MyOptions (assuming it has not already) or OptionsAck. Failure due to incompatible options or transmission problems must result in breaking the physical connection.

3.3.1.2 Protocol version exchange; the VersionReject and OptionsAck control packets

The purpose of the version number fields is to support the evolution of this protocol while allowing backward compatibility for implementations choosing to implement more than one version. True backward compatibility requires that the format of the MyOptions packet, up through the version number fields, remains constant forever.

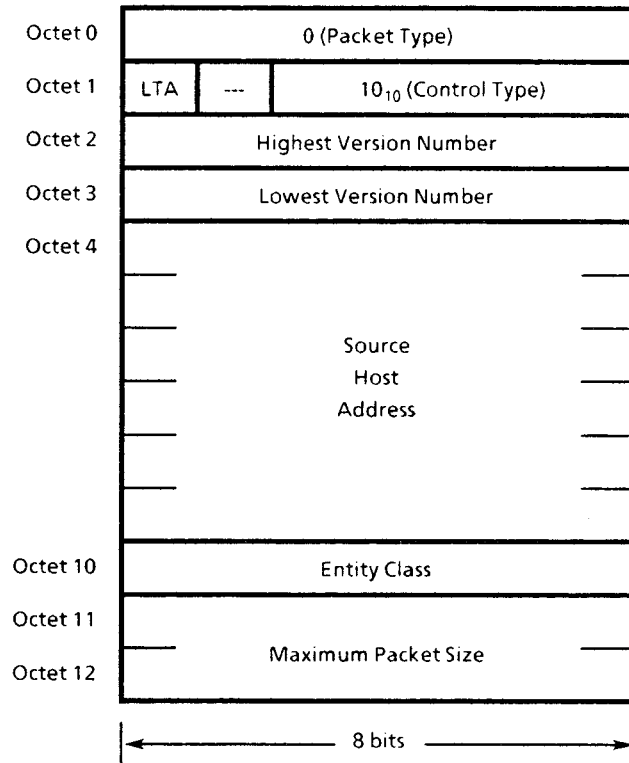


Figure 3.4 The MyOptions packet format

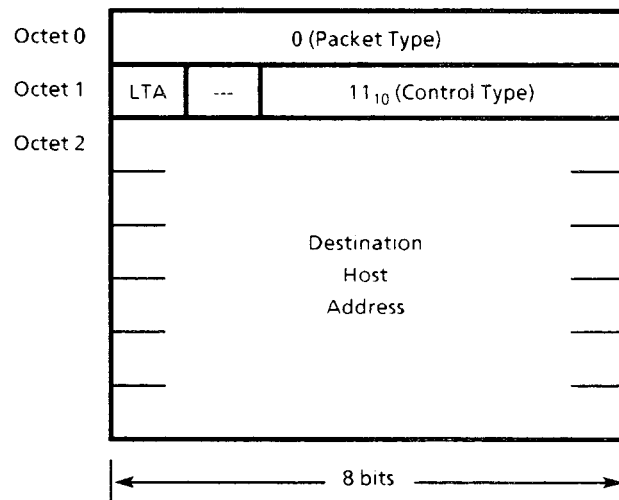


Figure 3.5 The OptionsAck packet format

The fields *lowest version number* and *highest version number* indicate the range, inclusively, of version numbers supported by the sender. The MyOptions packet is constructed using the format defined by the highest version field. If the receiver of the MyOptions packet supports the highest version and all other options are acceptable, it responds with an OptionsAck packet. Until there is agreement on the highest version number to be utilized, the rest of the fields in the MyOptions packet cannot be examined.

If the receiver of the MyOptions packet determines that the highest version number cannot be supported, it sends a VersionReject (Fig. 3.6) packet specifying the highest version number acceptable to it. If the mismatch is fatal, because the range of version numbers for each host has no intersection, the VersionReject packet concludes the negotiation and the physical connection is broken. If the mismatch is merely in the highest version, the

VersionReject packet specifies the highest intersecting version number. The recipient of this packet sends a new MyOptions packet echoing the highest intersecting version number.

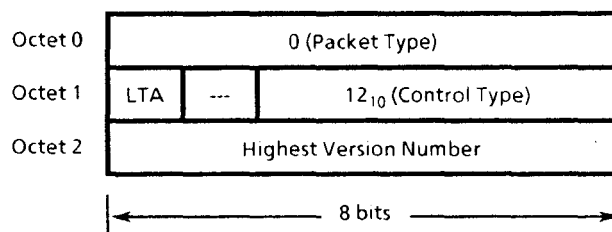


Figure 3.6 The VersionReject packet format

3.3.1.3 Source host address; the AddressReject control packet

The MyOptions packet holds a six-octet *source host address* field. Its encoding is identical to that specified in the Ethernet Specification [1]. When this address is specified, it indicates that the sending host possesses a unique host address. This address can be saved by the recipient and associated with all future packets that are received, as though the address accompanied each packet.

An anonymous host sends a MyOptions packet with a null source host address (all zeros). The recipient interprets the null address from a non-interfacing system element as a request for address assignment. If all other aspects of the option negotiation allow an OptionsAck packet to be returned, and the OptionsAck sender can supply a host address, the assigned address is returned in the OptionsAck packet. If the receiver of the MyOptions packet cannot supply a unique host address, it returns an AddressReject (Fig. 3.7) packet and breaks the physical connection.

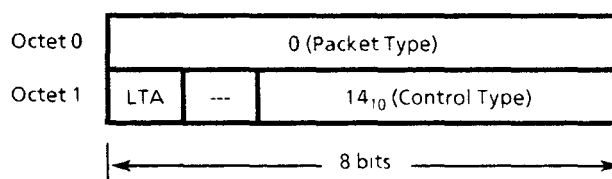


Figure 3.7 The AddressReject packet format

An Interfacing System Element (ISE) will use a null address as its source host address field in the MyOptions packets that it sends. An ISE will set its Ethernet receiver to the source host address in the MyOptions packet it receives, if that address is non-null. If the ISE supplies an address to an anonymous host, it will set its Ethernet receiver to the address it supplied to the anonymous host.

3.3.1.4 Entity class; the ClassReject control packet

An *entity class* field is included in the MyOptions packet to accommodate incompatibilities between communicating entities. The field is useful for determining illegal combinations of entities, such as between two interfacing elements or two cluster system elements. Defined values of entity class are Internetwork Router, Cluster System Element, Interfacing Element, and Terminal System Element. Table 3.2 shows the values assigned for entity class and the associations that are allowed between the hosts (the shaded areas in the table).

When an illegal combination of entities is encountered the host responds with a ClassReject packet (Fig 3.8).

Table 3.2 Allowed associations between entities

Shaded areas indicate a permissible association

	Entity Class Value	Inter-network Router	Cluster System Element	Interfacing System Element	Terminal System Element
Internetwork Router	0				
Cluster System Element	1				
Interfacing System Element	2				
Terminal System Element	3				

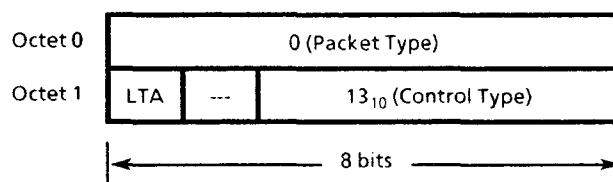


Figure 3.8 The ClassReject packet format

3.3.1.5 Maximum packet size; the SizeReject control packet

The protocol can be used to pass any Level 1 protocol packet of any length less than 65,536 octets. The maximum size of the packet depends on the implementations of the communicating entities. A *maximum packet size* field is included in the MyOptions packet to allow implementations to convey the maximum size of packets they will receive. The size is set independently for each direction. It specifies the maximum size of the packet in octets that can be received in one Synchronous packet.

If the recipient of a MyOptions packet cannot restrict its transmitted packets to the size specified in the maximum packet size, it returns a SizeReject packet (Fig. 3.9), and breaks the physical connection.

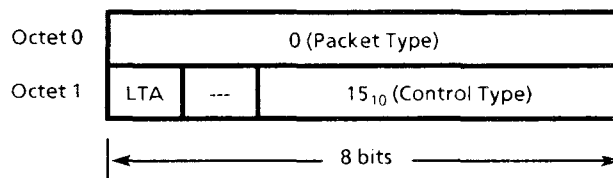


Figure 3.9 The SizeReject packet format

3.3.1.6 The Null control packet

The Null control packet is used primarily with half-duplex media to exchange control of the right-to-send (Fig. 3.10). It is used to send the line turn-around (LTA) flag when there is no accompanying data.

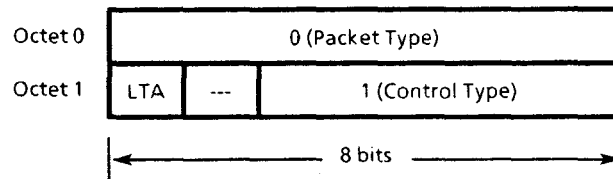


Figure 3.10 The Null packet format

3.3.1.7 The link check; the AreYouThere and IAmHere control packets

An entity may transmit an **AreYouThere** (Fig. 3.11) packet whenever it is in the data transfer phase (see section 4.1.3), and it suspects the remote entity is not active. An entity receiving an **AreYouThere** packet is obliged to respond with an **IAmHere** (Fig. 3.12) packet as soon as possible. No entity is forced to generate **AreYouThere** packets. If the Level 1 protocols are such that a regular flow of Level 1 packets is not guaranteed, the **AreYouThere** packet may be generated to check if a period of inactivity on the connection is to be attributed to a faulty remote entity or to an undetected broken physical connection.

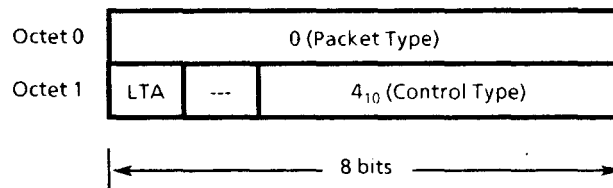


Figure 3.11 The AreYouThere packet format

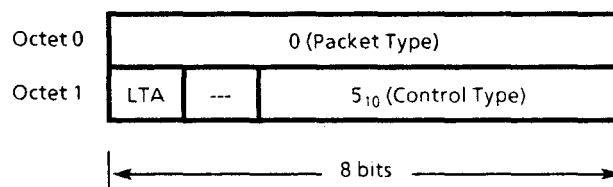


Figure 3.12 The IAmHere packet format

3.3.1.8 The TerminateRequest and TerminateReply control packets

The **TerminateRequest** and **TerminateReply** packets provide a reasonably reliable disconnect protocol (Fig. 3.13). The protocol is especially useful for switched media. It assumes that disconnecting with loss of Level 0 packets is acceptable. The disconnect can be initiated by either client, or by an inactivity timeout.

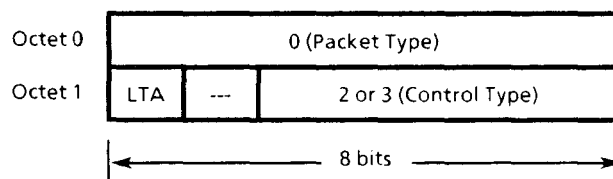


Figure 3.13 The Terminate packet formats

The initiator of the disconnect sends a `TerminateRequest` packet and sets a give-up timer. The initiator retransmits the `TerminateRequest` packet at fixed time intervals, up to the give-up timeout, or until a response is received. If a `TerminateReply` packet arrives or the give-up timeout occurs, the initiator breaks the physical connection (for example, goes “on-hook”). When simultaneous `TerminateRequest` packets are sent, each receiver responds with a `TerminateReply` packet, waits for a dally timeout, and then breaks the physical connection.

3.3.2 The Data packet

The Data packet (Fig. 3.14) is used to encapsulate Level 1 packets. The Level 1 packets are transported without regard for their contents. Data packets are identified by having a packet type in the range 1 to 255. The packet type value assists Level 1 implementations in the demultiplexing of packets when they are delivered via the Synchronous implementation’s service layer interface. The data packet types are defined in appendix C.

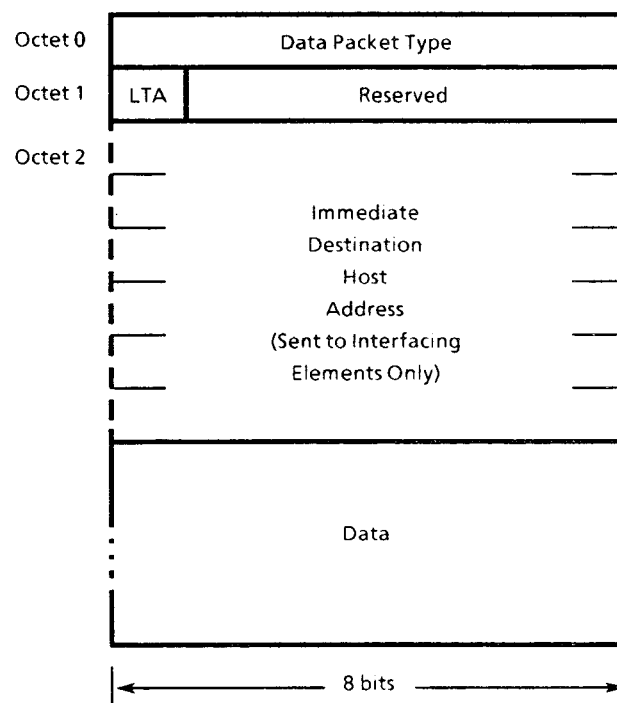


Figure 3.14 The Data packet format

The octet following the packet type is the *transmission-control* field. The first bit is for the control of line turn-around when using half-duplex media. The line turn-around (LTA) bit, when 1, indicates that the right to send on the half-duplex medium is passed to the receiver of the packet. See section 3.4 for more discussion of half-duplex line contention.

The remaining bits of the transmission control octet are reserved; no value is specified for them.

The *immediate-destination-host-address* field is a six octet host address, defined and encoded according to the Ethernet Specification. The host address specifies the host to which the Level 1 packet should be forwarded. It is present *only* when sending to an interfacing system element. Since the receiver of the packet is not itself acting as a host, the immediate destination address is required for forwarding. The Data field immediately follows the Reserved field when the receiving entity is not an interfacing system element.

The *data* field constitutes the remaining part of the data packet. It holds the Level 1 packet. Its length must be any integral number of octets, in the range of 0 to the maximum packet size indicated in the receiver's MyOptions packet.

3.3.3 Private packets

The protocol allows packet types that are private (Fig. 3.15). Private packets can have any format desired as long as the packet type is the first octet. The packet type field is centrally administered by Xerox (see appendix B). The formats for private packets are not part of this specification. An unrecognized private packet type should be ignored if received.

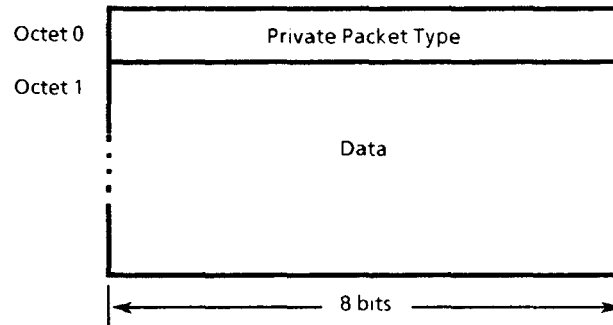


Figure 3.15 The Private packet format

3.4 Half-duplex medium contention

Half-duplex media can only support a single transmitter at a time. To manage contention for the right to transmit on the medium, the protocol defines a two-way alternation mode to be maintained between the communicating entities. There is also a master-slave relationship established. The host with the highest host address value is always the master. Since at most one host can specify a null host address, there is no ambiguity when dealing with an anonymous host. The anonymous host is always the slave.

When an ISE is connected to an anonymous host, the ISE will be the master. When an ISE is connected to a CE that possesses a host address, the ISE will be the slave since its host address is null.

The initial contention for master occurs during the option negotiation phase. As soon as the line becomes active, each host attempts to send a MyOptions packet. Once an entity learns of the other's host address, it assumes the proper role, otherwise it continues to vie for the line. Any collisions are resolved by each entity waiting a random time and then trying again.

Once the master-slave relationship is established, packets are sent in an alternating manner, using the LTA indicator to yield the right to send. For half-duplex to be reasonably efficient for random traffic, the medium must not be seized by a communicating entity. An entity must yield the right to send within three maximum-sized packet transmit times. If an entity has no data to send when given the right to send, it may delay before sending an LTA. The delay allows for "return" traffic from the entity's higher level clients. The delay is implementation dependent. Of course, the limitation on holding the right to send must include any such delay.

To protect against the loss of an LTA indication, the master acts as a watchdog. After having sent an LTA, if a master has not received an LTA from the slave within the timeout period defined below, the master transmits an LTA.

$$\text{Master watchdog timeout} = 1 + (3 \text{ packets} \times \text{MPS} \times 8 \text{ bits/octet}) / \text{LS}$$

MPS = maximum packet size in octets/packet

LS = linespeed in bits/second

Timeout values for *MPS* = 600 octets/packet follow:

<u>Linespeed (bps)</u>	<u>Timeout (secs.)</u>
1200	13.0
2400	6.5
4800	3.2
9600	1.6

3.5 Service interface

The Synchronous Point-to-Point Protocol provides the following operations to its Level 1 client. These operations constitute the service interface. This is not a protocol; these operations are guidelines for functionality only. They suggest an implementation interface style that need not be followed in actual implementations. The operations are based on an implementation model that supports a single physical interface and a single association.

Send: TYPE = PROCEDURE [levelOnePacket: Packet]

RETURNS []

REPORTS [noActivation, mediumDown, noAssociation];

Packet: TYPE = UNSPECIFIED;

Send passes a Level 1 packet to Level 0 for encapsulation and transmission on the line. The operation may fail if the client has not activated, the medium is not available, or the remote entity has not successfully negotiated an association.

Receive: TYPE = PROCEDURE []

RETURNS [levelOnePacket: Packet, status: ReceiveStatus]

REPORTS [noActivation, mediumDown, noAssociation];

ReceiveStatus: TYPE = {fcsError, packetAborted, associationReset, ...};

Receive consumes a Level 1 packet that was received and decapsulated by Level 0 and returns it in **levelOnePacket**. The Level 1 client is informed of any errors detected, such as an FCS error, by the Level 0 implementation through **status**.

GetStats: TYPE = PROCEDURE []

RETURNS [linkStats: LinkStats]

REPORTS [noActivation, mediumDown, noAssociation];

LinkStats: TYPE = RECORD [

state: ProtocolState,

perceivedLineSpeed: CARDINAL,

newAssociations: LONG CARDINAL,

```

octetsReceivedOK: LONG CARDINAL,
packetsReceivedOK: LONG CARDINAL,
octetsSentOK: LONG CARDINAL,
packetsSentOK: LONG CARDINAL,
unknownPacketTypeReceived: LONG CARDINAL,
fcsErrors: LONG CARDINAL,
receiveDataOverrun: LONG CARDINAL,
receiveDeviceError: LONG CARDINAL,
remoteNotResponding: LONG CARDINAL,
packetsToSendDiscardedDueToCongestion: LONG CARDINAL,
stuckOutput: LONG CARDINAL,
protocolErrors: LONG CARDINAL] ;

```

GetStats obtains information on the state of the transmission medium and Level 0 implementation. The statistics are cumulative. The client is responsible for keeping track of time and sampling the statistics to produce useful network management information and aids.

Activate: TYPE = PROCEDURE [role: ActivateRole]
 RETURNS []
 REPORTS [alreadyActive];

ActivateRole: TYPE = {active, passive};

Activate requests prepare the medium (communication facility) to become active. The **role** parameter indicates whether the Level 0 implementation should begin option negotiation or remain passive once the medium is available. Entities that use passive activation must use methods beyond the scope of this specification to prevent both entities from activating passively. If desired, both entities may use the active role.

Deactivate: TYPE = PROCEDURE []
 RETURNS []
 REPORTS [noActivation];

Deactivate requests the immediate termination of the Level 0 activation or association. Level 0 packets may be lost as a result of this operation; higher-level protocols are responsible for reliable data delivery.

AwaitMediumUp: TYPE = PROCEDURE []
 RETURNS []
 REPORTS [];

AwaitMediumUp waits for notification that the medium is available. The client may use this operation as an indication to perform **Activate**, **Send**, and **Receive** operations.

AwaitMediumDown: TYPE = PROCEDURE []
 RETURNS [reason: MediumDownReason]
 REPORTS [];

MediumDownReason: TYPE = {failedNegotiation, clientDeactivate, protocolError, remoteNotResponding};

AwaitMediumDown waits for notification that the medium is unavailable. This may be used as an indication to re-**Activate** in expectation of a new association.

The protocol defines the exchanges that may legally occur between communicating entities. It is possible to define the legal exchanges by describing the communicating entities as automata. Such a description is given in this chapter. The state machine is defined below as a series of states and the state transitions that result when various events occur.

4.1 The states of the automaton

The states are organized into four distinct phases: *idle*, *option negotiation*, *data transfer*, and *termination*. Events that cause state changes include: the arrival of frames, operations performed by Level 1 clients, and changes in the state of the physical medium. In the following discussion states are labeled by one letter followed by a number, events are labeled with two letters followed by a number.

A communicating entity (CE) is in the idle phase prior to physical connection to the medium or after physical disconnection from the medium. Physical connection corresponds to the medium interface equipment (for example, a modem) being available for the exchange of frames. A CE enters the option negotiation phase when a physical connection has been made and the Level 1 client has requested either an active or passive association to be established. An association is achieved when two CEs reach the data transfer phase. The CEs of an association enter the termination phase when one CE requests that the association be broken. Once the termination handshakes are complete, the physical connection is broken and both CEs return to the idle phase.

4.1.1 Idle phase

A CE in the idle phase is waiting for two events: a request by a Level 1 client to engage in an association, and an indication by the media interface equipment that transmission is possible. There are no restrictions on the order of these two events. An active (that is, willing to transmit) Level 1 client might first request the CE to make an association and subsequently arrange for the medium interface to be made ready. A passive (that is, willing to receive) Level 1 client might request an association only after receiving an indication that the medium interface is ready. Once both of the required events have occurred, the CE moves to the option negotiation phase.

A CE can only move from the idle phase to the option negotiation phase. A CE should always break the physical connection when it re-enters the idle phase. The idle phase can be entered from all other phases.

The idle phase contains the following states:

- I1.** Not Ready: neither the Level 1 client nor the medium is ready for an association.
- I2.** Awaiting Client: the medium interface has indicated that the medium is active and available for communication, but the client has not requested activation.
- I3.** Awaiting Physical Medium: the Level 1 client has requested an association, but the physical medium is not ready.

4.1.2 Option negotiation phase

A CE enters the option negotiation phase from the idle phase and remains in this phase until the negotiation succeeds or fails. (The process of option negotiation is discussed in detail in 3.3.1.1.) A successful negotiation moves a CE to the data transfer phase. There are several events that may cause a CE to return to the idle phase via the terminate phase, such as: unsuccessful negotiation, receipt of a request to terminate, problems with the medium, and unresponsiveness of the other CE.

The following states are defined in the option negotiation phase:

- O1.** Active Negotiation: entry into this state occurs when the Level 1 client activates with active negotiation, causing the CE to send a MyOptions packet.
- O2.** Passive Negotiation: entry into this state occurs if a Level 1 client activates with passive negotiation. The CE is prepared to negotiate options, but only after receiving a MyOptions packet. Only one CE can enter this state. Higher level conventions are required to prevent both entities from entering this state and deadlocking while they await the first MyOptions packet.
- O3.** Awaiting Options: The CE has received an OptionsAck to its own MyOptions packet and now awaits the arrival of the other CE's MyOptions packet.
- O4.** Awaiting OptionsAck: The CE has acknowledged the options of the other CE and has sent its own MyOptions. It retransmits a MyOptions packet until an acknowledgment, reject, or timeout occurs.

4.1.3 Data transfer phase

The data transfer phase is entered from the option negotiation phase only. Entry into this phase occurs after a CE has received and acknowledged a MyOptions packet, and when its own MyOptions packet has been acknowledged. Because an OptionsAck may be delayed or lost, entry into this state by one CE does not assure entry by the other. While in this state, a CE must acknowledge all MyOptions packets received that are identical to the original. This protects against lost OptionsAck packets.

This phase has only one state:

- D1.** Data Transfer: entry into this state means that an association has been established with the remote CE. Local Level 1 clients may now exchange packets with remote Level 1 clients.

If the connection is half-duplex then, when entities are in this state, they may be either sending or receiving, but not both in parallel. A change in the direction of data flow does not cause a change of state.

4.1.4 Termination phase

The termination phase normally is entered when either CE requests it via a `TerminateRequest` packet. This phase may be entered from the data transfer or option negotiation phases, and exit may only be made to the idle phase.

The following states are defined in the termination phase:

T1. Await TerminationReply: a `TerminateRequest` packet has been sent. The CE is awaiting a `TerminateReply` packet or a timeout prior to entering the idle phase. If a `TerminateRequest` packet is received, due to simultaneous termination of both CEs, a `TerminateReply` is sent and the Termination Dally state (T2) is entered.

T2. Termination Dally: a `TerminateReply` packet has been sent. Exit from this state is usually due to a timeout, though arrival of a `TerminateReply` is sufficient to move directly to the idle phase without waiting.

4.1.5 Loss of synchronization

Because packets may become lost or damaged at any time, it is possible to lose synchronization between the two Finite State Automata implemented by the communicating entities. Due to the nature of the protocol, loss of synchronization is only significant when it occurs during the option negotiation phase, preventing the CEs from reaching the data transfer phase.

During option negotiation, loss of synchronization will be detected as a protocol error or as a timeout. If this occurs, the CEs will break the connection. Breaking the physical connection will serve to re-synchronize the CEs at the idle phase, but only if they are both notified by the physical medium that the connection was broken.

DCEs that are connected to the Direct Distance Dial Network will usually detect loss of connection and notify the attached DTE (the CE) appropriately. DCEs that are connected to a leased circuit usually cannot detect loss of connection information from the distant DCE/DTE. Because of this, simply breaking the connection is insufficient to guarantee re-synchronization of the CEs.

CEs shall always execute the termination phase of the protocol before breaking the physical connection. This will insure re-synchronization at the idle phase.

4.2 Events

The protocol defines the transitions that occur when a state is occupied and a particular event occurs. The events that cause transitions are Level 1 client operations, the arrival of Level 0 frames (packets), timeouts, and state changes in the physical medium. Events will be described independently of the states. The interaction of states and events will be defined subsequently.

4.2.1 Client operations

The only operations considered here are those that might result in state transitions. The operations are described as procedure calls.

CO1. Activate[role]: the Level 1 client has requested that the implementation be prepared for the physical medium layer to become active. If **role** is **passive**, the CE should be passive with respect to option negotiation. If **role** is **active**, option negotiation should proceed when the physical medium is ready.

CO2. Deactivate[]: the client requests an immediate termination of the Level 0 association. The client is responsible for any Level 1 data that may be lost by such a request.

CO3. Send[level1Packet]: the client wishes to send a Level 1 packet, encapsulated in Level 0 framing.

4.2.2 Packet arrivals

The arrival of packets results in specific actions and state transitions, depending upon the current state of the protocol (see section 4.3). The event names corresponding to the arrival of the various packets are listed below.

PA1. MyOptions

PA2. OptionsAck

PA3. VersionReject

PA4. Remaining reject packets

PA5. TerminateRequest

PA6. TerminateReply

PA7. Data packet

PA8. Null or unspecified packet

PA9. AreYouThere

4.2.3 Medium status indications

ME1. Medium Up: The physical medium is available for transmission.

ME2. Medium Down: The physical medium is unavailable for transmission. Any active association is lost.

4.2.4 Timeouts

TO1. MyOptions Retransmit: No OptionsAck has been received in response to the last MyOptions. Suggested value: 2 seconds.

TO2. MyOptions Give-up: No OptionsAck has been received in response to any MyOptions. Suggested value: 20 seconds.

TO3. No Options: MyOptions packet was acknowledged, but no remote MyOptions packet arrived. Suggested value: 10 seconds.

TO4. Terminate Retransmit: A TerminateRequest was sent, but no TerminateReply was received; therefore the Terminate Request is retransmitted. Suggested value: 5 seconds.

TO5. Terminate Dally: A dally to allow transmission of TerminateReply before physical medium connection is broken. Suggested value: 2 seconds.

TO6. Terminate Giveup: A TerminateRequest was sent, but no TerminateReply was received. Suggested value: 20 seconds.

4.3 State transitions

Figures 4.1 and 4.2 give a general flowchart of the synchronous automaton. These flowcharts do not detail every possible transition; rather, they give the overall flow in the automaton.

Table 4.1 describes in detail the actions and transitions performed by each communicating entity in response to particular events while in a given state.

Interpretation of the matrix should be as follows. The states of the automaton are the rows, events are the columns. Each entry in the matrix indicates the action to be taken (an action is some activity that should occur before the state transition is made) and the next state. This is represented as "*action - next state*."

The correspondences between the action numbers and their meanings are given at the bottom of the table. State-event entries may have actions only, actions and next states, or next states only. States that are blank indicate that no action is necessary and no state change is necessary.

Two error actions are specified for unexpected events: local errors and remote errors. The first refers to unexpected client or local implementation behavior. These should be handled in an implementation-dependent manner. The second refers to loss of state synchronization with the remote entity. Recovery from loss of state synchronization consists of logging the event, notifying the client, completing the terminate phase, breaking the physical connection, and resynchronizing to the idle phase. (See section 4.1.5.)

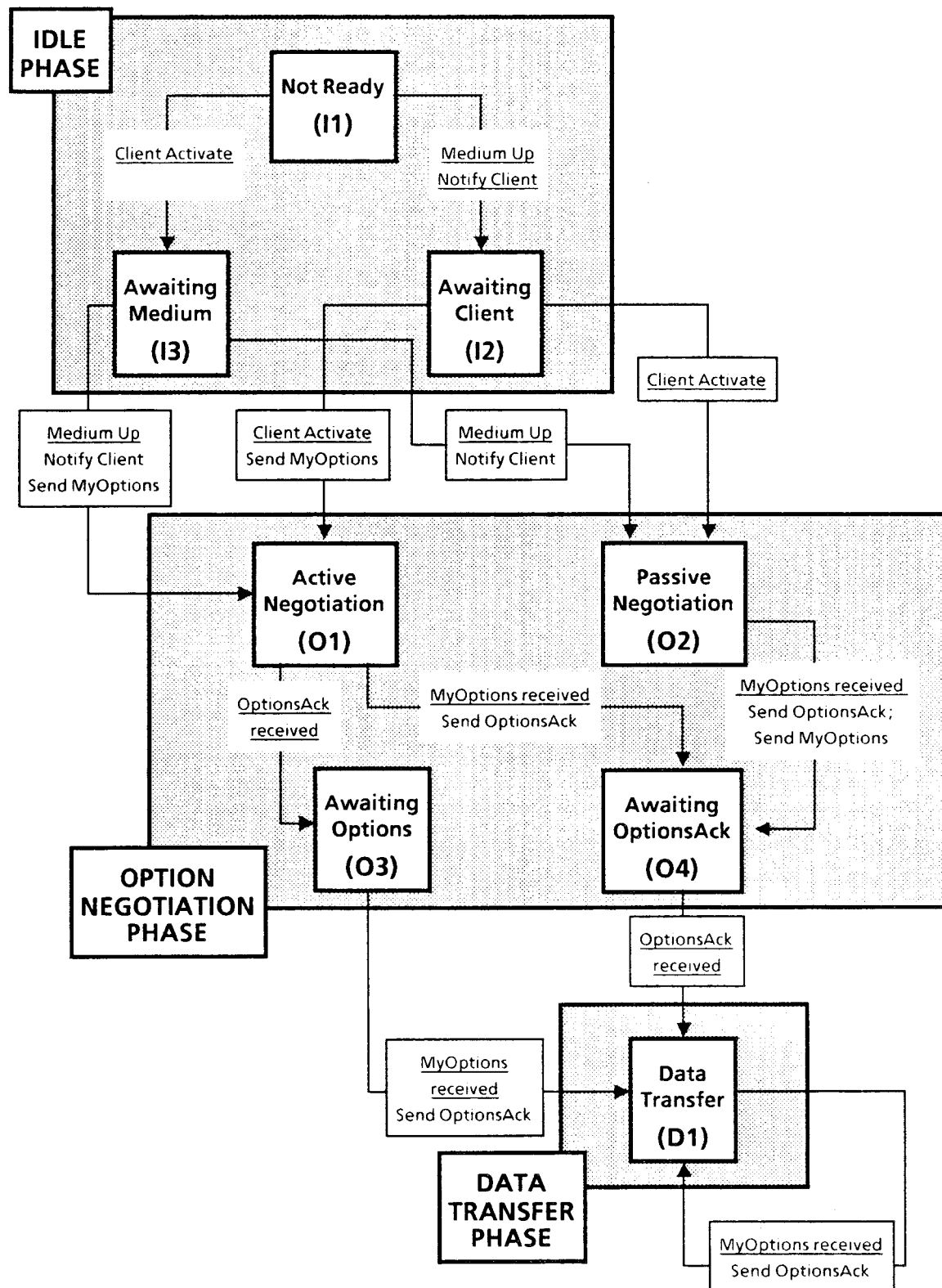


Figure 4.1 Activating states

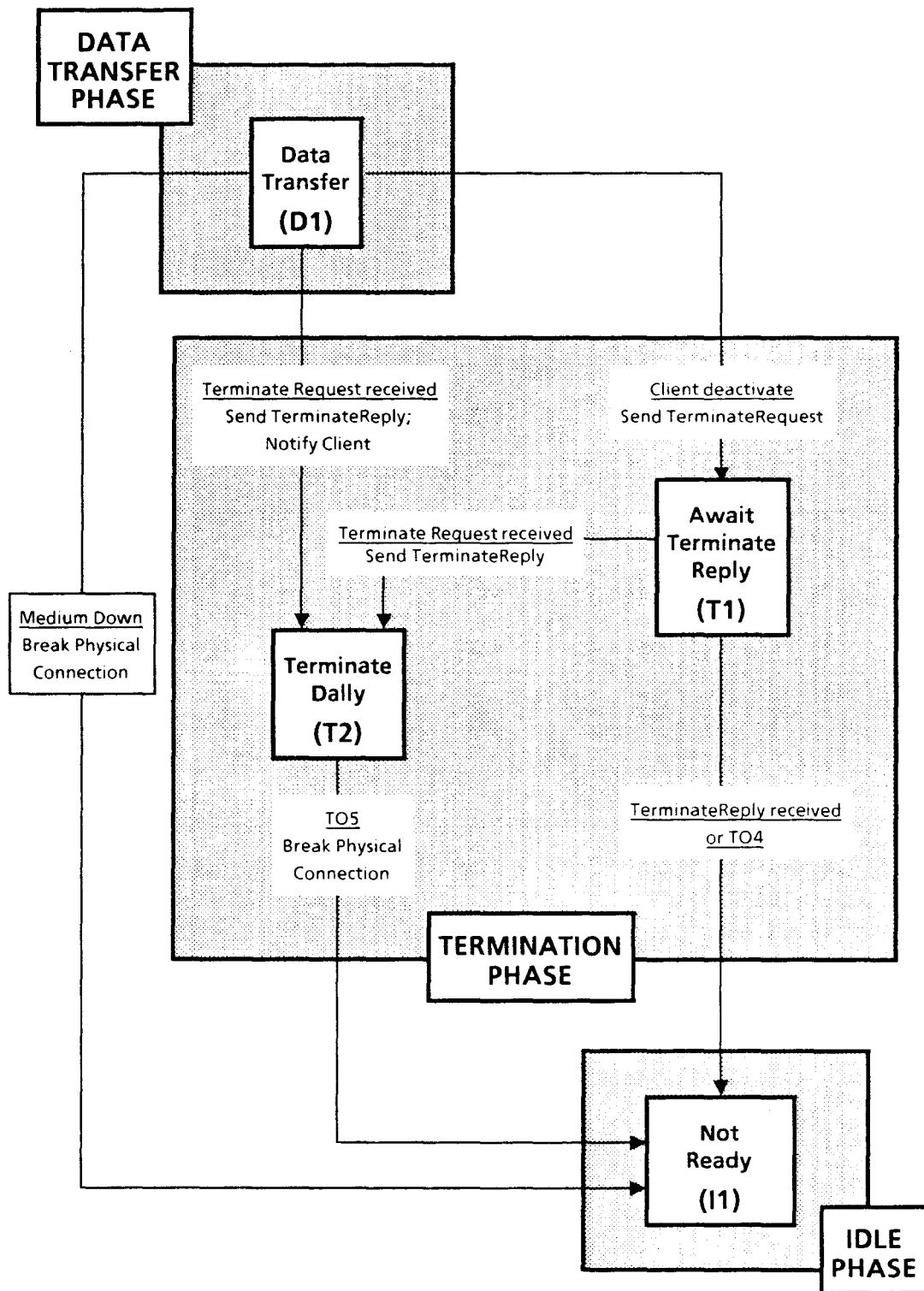


Figure 4.2 Deactivating states

Table 4.1 The Synchronous Point-to-Point State Table

EVENTS																						
	Phase	State	Client Operations			Medium		Packets									Timeouts					
			CO1	CO2	CO3	ME1	ME2	PA1	PA2	PA3	PA4	PA5	PA6	PA7	PA8	PA9	TO1	TO2	TO3	TO4	TO5	TO6
S T A T E S	IDLE	I1	I3	q-	q-	a-I2	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-
		I2	NB1	I1	q-	q-	a-I1	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-
		I3	q-	I1	q-	NB2	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-	q-
	OPTIONS	O1	q-	j-T1	q-	q-	a-I1	NB3	O3	NB4	j-T1	h-T2	rj-T1	rj-T1		rj-T1	f-	j-T1	q-	q-	q-	q-
		O2	q-	j-T1	q-	q-	a-I1	NB5	rj-T1	rj-T1	rj-T1	h-T2	rj-T1	rj-T1		rj-T1	q-	q-	q-	q-	q-	q-
		O3	q-	j-T1	q-	q-	a-I1	NB6		rj-T1	rj-T1	h-T2	rj-T1	rj-T1		rj-T1	q-	q-	j-T1	q-	q-	q-
		O4	q-	j-T1	q-	q-	a-I1	c-	D1	NB4	rj-T1	h-T2	rj-T1	f-	f-	f-	f-	j-T1	q-	q-	q-	q-
	DATA	D1	q-	j-T1	l-	q-	a-I1	c-		rj-T1	rj-T1	h-T2	rj-T1	a-		k-		q-	q-	q-	q-	q-
	TERMIN- ATION	T1	q-	q-	q-	q-	a-I1	rj-	rj-	rj-	rj-	e-T2	b-I1				q-	q-	q-	j-	q-	b-I1
		T2	q-	q-	q-	q-	a-I1	rj-	rj-	rj-	rj-						q-	q-	q-	q-	b-I1	q-

Actions

a - Notify Client
b - Break Connection
c - Send OptionsAck

d - Send MyOptions with highest version
e - Send TerminateReply
f - Send MyOptions

g - Send appropriate reject
h - Notify Client & send TerminateReply
i - Notify Client & break connection

j - Send TerminateRequest
k - Send IAmHere
l - Queue the packet to be sent

Error Actions

q - local error (implementation dependent action)

r - remote error: Notify Client

States

I1: Not Ready
I2: Awaiting Client

I3: Awaiting Physical Medium
O1: Active Negotiation

O2: Passive Negotiation
O3: Awaiting Options

O4: Awaiting OptionsAck
D1: Data Transfer

T1: Await TerminationReply
T2: Termination Dally

Events

CO1: **Activate[]**
CO2: **Deactivate[]**
CO3: **SendPacket[]**
ME1: Medium Up

ME2: Medium Down
PA1: MyOptions
PA2: OptionsAck
PA3: VersionReject

PA4: other rejects
PA5: TerminateRequest
PA6: TerminateReply
PA7: Data

PA8: Null
PA9: AreYouThere
TO1: MyOptions Retransmit
TO2: OptionsAck Giveup

TO3: No MyOptions
TO4: Terminate Retransmit
TO5: Terminate Dally
TO6: Terminate Giveup

Notes

NB1: If role = passive THEN O2 ELSE f-O1
NB2: If role = passive THEN a-O2 ELSE af-O1
NB3: If received options compatible THEN c-O4 ELSE g-

NB4: If version ranges intersect THEN d- ELSE j-T1
NB5: If received options compatible THEN cf-O4 ELSE g-
NB6: If received options compatible THEN c-D1 ELSE g-

The following documents supplement this protocol specification.

- [1] Digital Equipment Corporation; Intel Corporation; Xerox Corporation. *The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications*. November 1982; Version 2.0.

This reference contains the data link and physical layer specifications for the Ethernet.

- [2] Electronic Industries Association. *EIA RS-232-C Standard: Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange*. August 1969.

This reference defines the RS-232-C protocol.

- [3] International Business Machines. *IBM Synchronous Data Link Control*. March 1979. Document No. GA27-3093-2.

This reference defines the SDLC protocol.

- [4] ISO International Standard (IS) 3309, *Data Communications-High-Level Data Link Control Procedures-Frame Structure*; 1979.

This reference defines the HDLC framing conventions.

- [5] Xerox Corporation. *Courier: The Remote Procedure Call Protocol*. Xerox Network Systems Standard. Stamford, Connecticut; December 1981. XNSS 038112 (XSIS 038112).

This reference defines the Courier Protocol and its data typing conventions and syntax.

- [6] Xerox Corporation. *Internet Transport Protocols*. Xerox Network Systems Standard. Stamford, Connecticut; December 1981. XNSS 028112 (XSIS 028112).

This reference defines the NS Internetwork Datagram Protocol.

- [7] Xerox Corporation, *Pup: An Internetwork Architecture*, Palo Alto Research Center, July 1979. CSL-79-10.

This reference defines the Pup protocol.

B.

As stated in this document, packet types are assigned 8-bit numbers that are unique throughout the distributed system. This number space is administered by Xerox Corporation. To obtain numbers for use as packet types, submit a written request to:

Xerox Corporation
Xerox Network Systems Institute
2300 Geng Road
Palo Alto, California 94303

Protocol implementors are encouraged to apply for unique blocks of numbers for their particular applications. Uniqueness allows systems to freely interconnect without having to worry about overlapping values for critical fields.

Type numbers should be used with economy as the total number of blocks is limited. If an implementation is using a large quantity of these type numbers, the designer has probably misunderstood their purpose.

C. ASSIGNED PACKET TYPES

As stated in this document, packet types are assigned 8-bit numbers that are unique throughout the distributed system. The following packet types have been assigned:

Table C.1 Packet types

Packet Type (decimal)	Function
0	Control packet
4	Pup packet
12	NS Internet Datagram
13	U.S. Dept. of Defense MIL-STD-1777, and DARPA RFC-791, Internet Protocol
14	ISO 8473 Data Communications Protocol
16	Set SIU (private packet)
17	SIU Echo packet (private packet)
18	SIU Determine Line Speed (private packet)
240- 255	Experimental

Xerox Corporation
Stamford, Connecticut 06904

XEROX® is a trademark of
XEROX CORPORATION.

Printed in U.S.A. OSD 001(11/82)