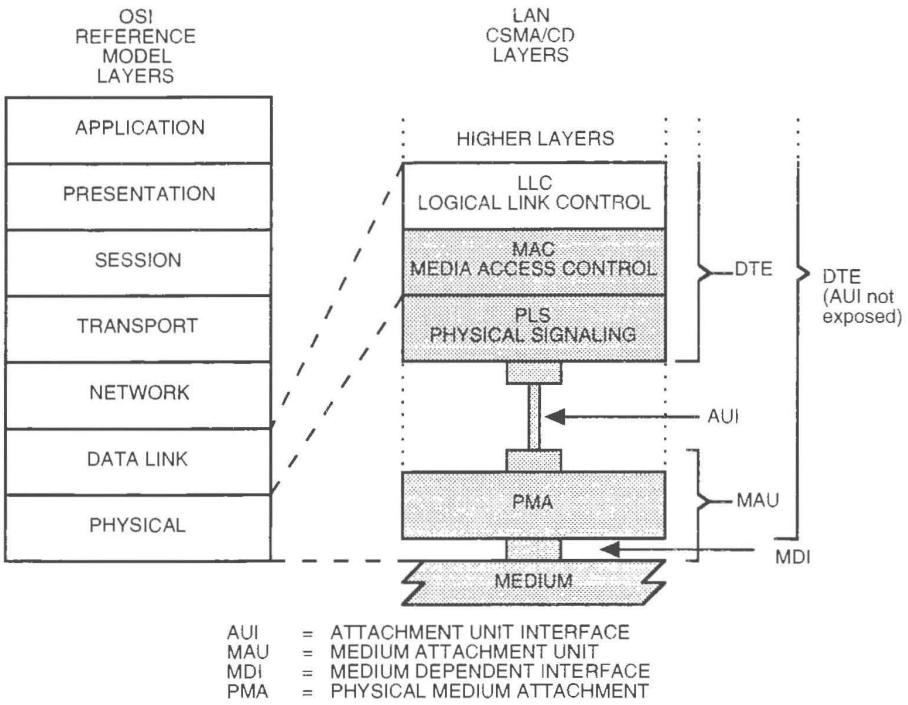


Fig 1-1
LAN Standard Relationship to the ISO Open Systems Interconnection (OSI) Reference Model

The architectural model is based on a set of interfaces that may be different from those emphasized in implementations. One critical aspect of the design, however, shall be addressed largely in terms of the implementation interfaces: compatibility.

1.1.2.2 Two important compatibility interfaces are defined within what is architecturally the Physical Layer.

(1) *Medium-Dependent Interface (MDI).* To communicate in a compatible manner, all stations shall adhere rigidly to the exact specification of physical media signals defined in Section 8 (and beyond) in this standard, and to the procedures that define correct behavior of a station. The medium-independent aspects of the LLC sublayer and the MAC sublayer should not be taken as detracting from this point; communication by way of the ISO 8802-3 [IEEE 802.3] Local Area Network requires complete compatibility at the Physical Medium interface (that is, the coaxial cable interface).

(2) *Attachment Unit Interface (AUI).* It is anticipated that most DTEs will be located some distance from their connection to the coaxial cable. A small amount of circuitry will exist in the Medium Attachment Unit (MAU) directly adjacent to the coaxial cable, while the majority of the hardware and all of the software will be placed within the DTE. The AUI is defined as a second compatibility interface. While conformance with this interface is not strictly necessary to ensure communication, it is highly recommended, since it allows maximum flexibility in intermixing MAUs and DTEs. The AUI may be optional or not specified for some implementations of this standard that are expected to be connected directly to the medium and so do not use a separate MAU or its interconnecting AUI cable. The PLS and PMA are then part of a single unit, and no explicit AUI specification is required.

1.1.3 Layer Interfaces. In the architectural model used here, the layers interact by way of well defined interfaces, providing services as specified in Sections 2 and 6. In general, the interface requirements are as follows.

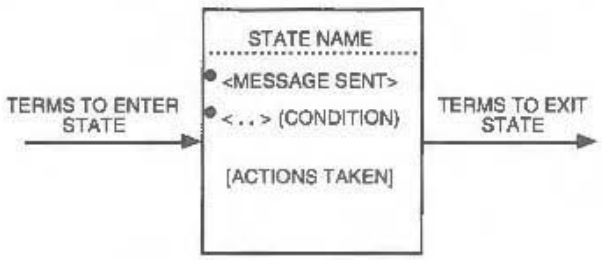
(1) The interface between the MAC sublayer and the LLC sublayer includes facilities for transmitting and receiving frames, and provides per-operation status information for use by higher-layer error recovery procedures.
(2) The interface between the MAC sublayer and the Physical Layer includes signals for framing (carrier sense, transmit initiation) and contention resolution (collision detect), facilities for passing a pair of serial bit streams (transmit, receive) between the two layers, and a wait function for timing.

These interfaces are described more precisely in 4.3. Additional interfaces are necessary to allow higher level network management facilities to interact with these layers to perform operation, maintenance, and planning functions. Network management functions will be discussed in Section 5.

1.1.4 Application Areas. The applications environment for the Local Area Network is intended to be commercial and light industrial. Use of CSMA/CD LANs in home or heavy industrial environments, while not precluded, is not considered within the scope of this standard.

1.2 Notation

1.2.1 State Diagram Conventions. The operation of a protocol can be described by subdividing the protocol into a number of interrelated functions. The operation of the functions can be described by state diagrams. Each diagram represents the domain of a function and consists of a group of connected, mutually exclusive states. Only one state of a function is active at any given time (see Fig 1-2).



Key: ( ) = condition, for example, (if no_collision)
[ ] = action, for example, [reset PLS functions]
* = logical AND
+ = logical OR
Tw = Wait Time, Implementation dependent
Td = Delay Timeout
Tb = Backoff Timeout
UCT = unconditional transition

Fig 1-2
State Diagram Notation Example

Each state that the function can assume is represented by a rectangle. These are divided into two parts by a horizontal line. In the upper part the state is identified by a name in capital letters. The lower part contains the name of any ON signal that is generated by the function. Actions are described by short phrases and enclosed in brackets.

All permissible transitions between the states of a function are represented graphically by arrows between them. A transition that is global in nature (for example, an exit condition from all states to the IDLE or RESET state) is indicated by an open arrow. Labels on transitions are qualifiers that must be fulfilled before the transition will be taken. The label UCT designates an unconditional transition. Qualifiers described by short phrases are enclosed in parentheses.

State transitions and sending and receiving of messages occur instantaneously. When a state is entered and the condition to leave that state is not immediately fulfilled, the state executes continuously, sending the messages and executing the actions contained in the state in a continuous manner.

Some devices described in this standard (e.g., repeaters) are allowed to have two or more ports. State diagrams capable of describing the operation of devices with an unspecified number of ports, required qualifier notation that allows testing for conditions at multiple ports. The notation used is a term that includes a description in parentheses of which ports must meet the term for the qualifier to be satisfied (e.g., ANY and ALL). It is also necessary to provide for term-assignment statements that assign a name to a port that satisfies a qualifier. The following convention is used to describe a term-assignment statement that is associated with a transition:

(1) The character ":" (colon) is a delimiter used to denote that a term assignment statement follows.
(2) The character "⇐" (left arrow) denotes assignment of the value following the arrow to the term preceding the arrow.

The state diagrams contain the authoritative statement of the functions they depict; when apparent conflicts between descriptive text and state diagrams arise, the state diagrams are to take precedence. This does not override, however, any explicit description in the text that has no parallel in the state diagrams.

The models presented by state diagrams are intended as the primary specifications of the functions to be provided. It is important to distinguish, however, between a model and a real implementation. The models are optimized for simplicity and clarity of presentation, while any realistic implementation may place heavier emphasis on efficiency and suitability to a particular implementation technology. It is the functional behavior of any unit that must match the standard, not its internal structure. The internal details of the model are useful only to the extent that they specify the external behavior clearly and precisely.

1.2.2 Service Specification Method and Notation. The service of a layer or sublayer is the set of capabilities that it offers to a user in the next higher (sub)layer. Abstract services are specified here by describing the service primitives and parameters that characterize each service. This definition of service is independent of any particular implementation (see Fig 1-3).

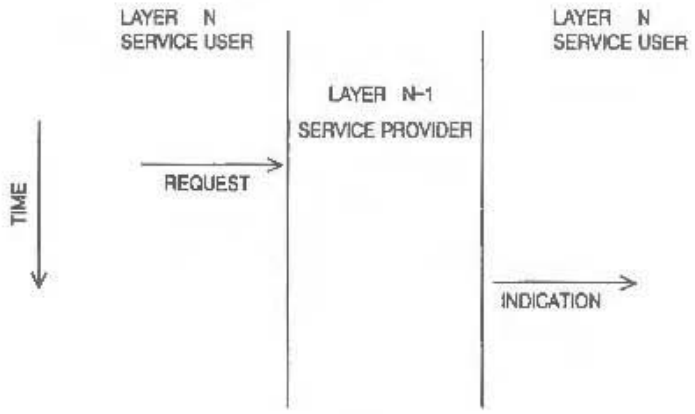


Fig 1-3
Service Primitive Notation

Specific implementations may also include provisions for interface interactions that have no direct end-to-end effects. Examples of such local interactions include interface flow control, status requests and indications, error notifications, and layer management. Specific implementation details are omitted from this service specification both because they will differ from implementation to implementation and because they do not impact the peer-to-peer protocols.

1.2.2.1 Classification of Service Primitives. Primitives are of two generic types:

(1) REQUEST. The request primitive is passed from layer N to layer N-1 to request that a service be initiated.

(2) INDICATION. The indication primitive is passed from layer N-1 to layer N to indicate an internal layer N-1 event that is significant to layer N. This event may be logically related to a remote service request, or may be caused by an event internal to layer N-1.

The service primitives are an abstraction of the functional specification and the user-layer interaction. The abstract definition does not contain local detail of the user/provider interaction. For instance, it does not indicate the local mechanism that allows a user to indicate that it is awaiting an incoming call. Each primitive has a set of zero or more parameters, representing data elements that shall be passed to qualify the functions invoked by the primitive. Parameters indicate information available in a user/provider interaction; in any particular interface, some parameters may be explicitly stated (even though not explicitly defined in the primitive) or implicitly associated with the service access point. Similarly, in any particular protocol specification, functions corresponding to a service primitive may be explicitly defined or implicitly available.

1.2.3 Physical Layer and Media Notation. Users of this standard need to reference which particular implementation is being used or identified. Therefore, a means of identifying each implementation is given by a simple, three-field, type notation that is explicitly stated at the beginning of each relevant section. In general, the Physical Layer type is specified by these fields:

<data rate in Mb/s> <medium type> <maximum segment length (× 100 m)>

For example, the standard contains a 10 Mb/s baseband specification identified as "TYPE 10BASE5," meaning a 10 Mb/s baseband medium whose maximum segment length is 500 m. Each successive Physical Layer specification will state its own unique TYPE identifier along similar lines.

1.2.4 Physical Layer Message Notation. Messages generated within the Physical Layer, either within or between PLS and the MAU (that is, PMA circuitry), are designated by an italic type to designate either form of physical or logical message used to execute the physical layer signaling process (for example, *input_idle* or *mau_available*).

1.3 References

[1] CISPR Publication 22 (1985), Limits and Methods of Measurement of Radio Interference Characteristics of Information Technology Equipment.[2]

[2] IEC Publication 96-1, Radio-frequency cables, Part 1: General requirements and measuring methods.[3]

[3] IEC Publication 96-1A, 1st Supplement to Radio-frequency cables, Part 1: Appendix Section 5.4, Terminated triaxial test method for transfer impedance up to 100 MHz.

[4] IEC Publication 169-8 and -16, Radio-frequency connectors, Part 8: Radio-frequency coaxial connectors with inner diameter of outer conductor 6.5 mm (0.256 in) with bayonet lock—Characteristic impedance 50 ohms (Type BNC); Part 16: Radio-frequency coaxial connectors with inner diameter of outer conductor 7 mm (0.276 in) with screw coupling—Characteristic impedance 50 ohms (75 ohms) (Type N).

[5] IEC Publication 380, Safety of electrically energized office machines.

[6] IEC Publication 435, Safety of data processing equipment.

[2]CISPR documents are available from the International Electrotechnical Commission, 3 rue de Varembé, Case Postale 131, CH 1211, Genève 20, Switzerland/Suisse. CISPR documents are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[3]IEC publications are available from IEC Sales Department, Case Postale 131, 3 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[7] IEC Publication 807-2, Rectangular connectors for frequencies below 3 MHz, Part 2: Detail specification for a range of connectors with round contacts—Fixed solder contact types.

[8] IEC Publication 950, Safety of Information Technology Equipment, Including Electrical Business Equipment.

[9] ISO 2382-9 : 1984, Data processing—Vocabulary—Part 09: Data communication.[4]

[10] ISO 7498 : 1984, Information processing systems–open systems interconnection–Basic reference model.

[11] IEC Publication 60, High-voltage test techniques.

[12] IEC Publication 68, Environmental testing.

[13] IEC Publication 793-1, Optical fibres, Part 1: Generic specification.

[14] IEC Publication 793-2, Optical fibres, Part 2: Product specifications.[5]

[15] IEC Publication 794-1, Optical fibre cables, Part 1: Generic specification.

[16] IEC Publication 794-2, Optical fibres cables, Part 2: Product specifications.

[17] IEC Publication 825, Radiation safety of laser products, equipment classification, requirements, and user's guide.

[18] IEC Publication 874-1, Connectors for optical fibres and cables, Part 1: Generic specification.

[19] IEC Publication 874-2, Connectors for optical fibres and cables, Part 2: Sectional specification for fibre optic connector type F-SMA.

[20] ISO/IEC 7498-4 : 1989, Information processing systems—Open Systems Interconnection—Basic Reference Model—Part 4: Management Framework.

[21] ISO 8877 : 1987, Information processing systems—Interface connector and contact assignments for ISDN basic access interface located at reference points S and T.

Local and national standards such as those supported by ANSI, EIA, IEEE, MIL, NFPA, and UL are not a formal part of the ISO/IEC 8802-3 standard. Reference to such local or national standards may be useful resource material and are identified by a bracketed number beginning with the letter A and located in Annex A.

1.4 Definitions. The definitions used in this standard are consistent with ISO 2382-9:1984 [9]. A more specific Part 25, pertaining to LAN systems, is in development.

---

[4]ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[5] Subsection 9.9 is to be read with the understanding that the following changes to IEC Publication 793-2 [14] have been requested:
(1) Correction of the numerical aperture tolerance in Table III to ± 0.015.
(2) Addition of another bandwidth category, of ≥ 150 MHz referred to 1 km, for the type A1b fiber in Table III.

2. MAC Service Specification

2.1 Scope and Field of Application. This section specifies the services provided by the Media Access Control (MAC) sublayer to the Logical Link Control (LLC) sublayer for the ISO [IEEE] Local Area Network standard (see Fig 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 4.2 and 4.3.

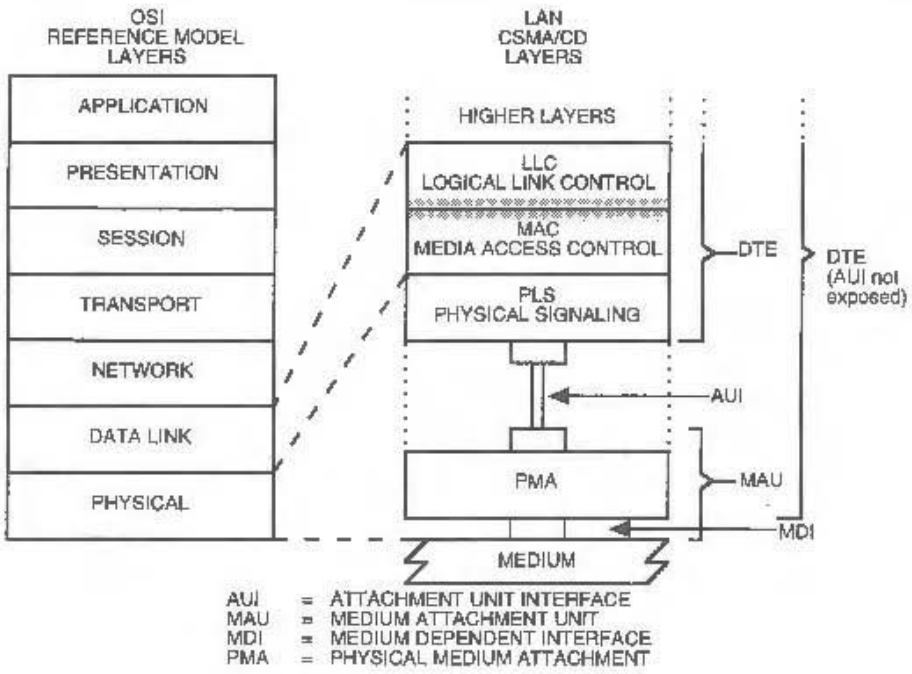


Fig 2-1
Service Specification Relation to the LAN Model

2.2 Overview of the Service

2.2.1 General Description of Services Provided by the Layer. 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.

2.2.2 Model Used for the Service Specification. The model used in this service specification is identical to that used in 1.2.

2.2.3 Overview of Interactions

MA_DATA.request
MA_DATA.indication

2.2.4 Basic Services and Options. The MA_DATA.request and MA_DATA.indication service primitives described in this section are considered mandatory.

2.3 Detailed Service Specification

2.3.1 MA_DATA.request

2.3.1.1 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 in the case of group addresses.

2.3.1.2 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 and any physical 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 2.3.1.5).

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

2.3.1.4 Effect of Receipt. The receipt of this primitive will cause 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 pass the properly formed frame to the lower protocol layers for transfer to the peer MAC sublayer entity or entities.

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

2.3.2 MA_DATA.indication

2.3.2.1 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.

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

```
MA_DATA.indication  (
                    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 source_address parameter is 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 is used to pass status information to the peer LLC sublayer entity.

2.3.2.3 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 to 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.

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

2.3.2.5 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 (for example, all frames transmitted to the broadcast address will invoke MA_DATA.indication at all stations in the network including the station that generated the request).

3. Media Access Control Frame Structure

3.1 Overview. This section defines in detail the frame structure for data communication systems using local area network 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 multistation) addresses, and into user administered and globally administered addresses.

3.1.1 MAC Frame Format. Figure 3-1 shows the eight fields of a frame: the preamble, Start Frame Delimiter (SFD), 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 errors in received frames. Of these eight fields, all are of fixed size except the LLC data and PAD fields, which may contain any integer number of octets between the minimum and maximum values determined by the specific implementation of the CSMA/CD Media Access mechanism. See 4.4 for particular implementations.

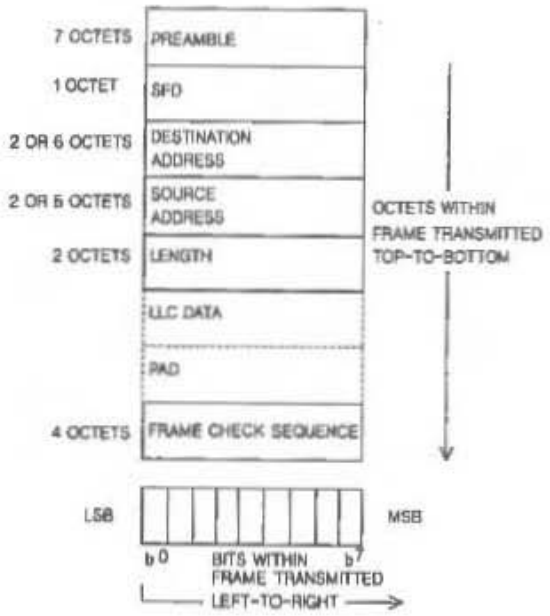


Fig 3-1
MAC Frame Format

The minimum and maximum frame size limits in 4.4 refer to that portion of the frame from the destination address field through the frame check sequence field, inclusive.

Relative to Fig 3-1, the octets of a frame are transmitted from top to bottom, and the bits of each octet are transmitted from left to right.

3.2 Elements of the MAC Frame

3.2.1 Preamble Field. The preamble field is a 7-octet field that is used to allow the PLS circuitry to reach its steady-state synchronization with the received frame timing (see 4.2.5).

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

3.2.3 Address Fields. Each MAC frame shall contain two address fields: the Destination Address field and the Source Address field, in that order. The Destination Address field shall specify the destination addressee(s) for which the frame is intended. The Source Address field shall identify the station from which the frame was initiated. The representation of each address field shall be as follows (see Fig 3-2):

(1) Each address field shall contain either 16 bits or 48 bits. However, at any given time, the Source and Destination Address size shall be the same for all stations on a particular local area network.
(2) The support of 16 or 48 bit address length for Source and Destination Address shall be left to the manufacturer as an implementation decision. There is no requirement that manufacturers support both sizes.
(3) The first bit (LSB) shall 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. In the Source Address field, the first bit is reserved and set to 0. If this bit is 0, it shall indicate that the address field contains an individual address. If this bit is 1, it shall 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.
(4) For 48 bit addresses, the second bit shall be used to distinguish between locally or globally administered addresses. For globally administered (or U, universal) addresses, the bit is set to 0. If an address is to be assigned locally, this bit shall be set to 1. Note that for the broadcast address, this bit is also a 1.
(5) Each octet of each address field shall be transmitted least significant bit first.

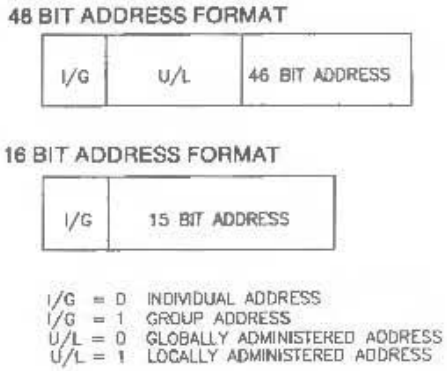


Fig 3-2
Address Field Format

3.2.3.1 Address Designation. A MAC sublayer address is of one of two types:

(1) *Individual Address.* The address associated with a particular station on the network.
(2) *Group Address.* A multidestination address, associated with one or more stations on a given network. There are two kinds of multicast address:
   (a) *Multicast-Group Address.* An address associated by higher-level convention with a group of logically related stations.
   (b) *Broadcast Address.* A distinguished, predefined multicast address that 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 LANs) shall be predefined to be the Broadcast address. This group shall be predefined for each communication medium to consist of all stations

actively connected to that medium; it shall be used to broadcast to all the active stations on that medium. All stations shall be able to recognize the Broadcast address. It is not necessary that a station be capable of generating the Broadcast address.

The address space shall also be partitioned into locally administered and globally administered addresses. The nature of a body and the procedures by which it administers these global (U) addresses is beyond the scope of this standard.[6]

3.2.4 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.

3.2.5 Source Address Field. The Source Address field specifies the station sending the frame. The Source Address field is not interpreted by the CSMA/CD MAC sublayer.

3.2.6 Length Field. The length field is a 2-octet field whose value[7] 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 field (a sequence of octets) will be added at the end of the data field but prior to the FCS field, specified below. The procedure that determines the size of the pad field is specified in 4.2.8. The length field is transmitted and received with the high order octet first.

3.2.7 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 number specified by the implementation of this standard that is used. A minimum frame size is required for correct CSMA/CD protocol operation and is specified by the particular implementation of the standard. If necessary, the data field is extended by appending extra bits (that is, a pad) in units of octets after the LLC data field but prior to calculating and appending the FCS. The size of the pad, if any, is determined by the size of the data field supplied by LLC and the minimum frame size and address size parameters of the particular implementation. The maximum size of the data field is determined by the maximum frame size and address size parameters of the particular implementation.

The length of PAD field required for LLC data that is n octets long is max (0, minFrameSize – (8 × n + 2 × addressSize + 48)) bits. The maximum possible size of the LLC data field is maxFrameSize – (2 × addressSize + 48)/8 octets. See 4.4 for a discussion of implementation parameters; see 4.2.3.3 for a discussion of the minFrameSize.

3.2.8 Frame Check Sequence Field. A cyclic redundancy check (CRC) is used by 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 address, destination address, length, LLC data and pad (that is, all fields except the preamble, SFD, and FCS). The encoding is defined by the following generating polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

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

(1) The first 32 bits of the frame are complemented.
(2) 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 $x^{(n-1)}$ term and the last bit of the data field corresponds to the x^0 term.)
(3) M(x) is multiplied by x^{32} and divided by G(x), producing a remainder R(x) of degree <31.
(4) The coefficients of R(x) are considered to be a 32-bit sequence.
(5) The bit sequence is complemented and the result is the CRC.

[6] For information on how to use MAC addresses, see IEEE Std 802-1990, Overview and Architecture. To apply for an Organizationally Unique Identifier for building a MAC address, contact the Registration Authority, IEEE Standards Department, P.O. Box 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA; (908) 562-3813; fax (908) 562-1571.

[7] Packets with a length field value greater than those specified in 4.4.2 may be ignored, discarded, or used in a private manner. The use of such packets is beyond the scope of this standard.

The 32 bits of the CRC value are placed in the frame check sequence field so that the x^{31} term is the left-most bit of the first octet, and the x^0 term is the right most bit of the last octet. (The bits of the CRC are thus transmitted in the order $x^{31}, x^{30}, \ldots, x^1, x^0$.) See reference [A18].

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

3.4 Invalid MAC Frame. An invalid MAC frame shall be defined as one that meets at least one of the following conditions:

(1) The frame length is inconsistent with the length field.
(2) It is not an integral number of octets in length.
(3) 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 shall not be passed to LLC. The occurrence of invalid MAC frames may be communicated to network management.

4. Media Access Control

4.1 Functional Model of the Media Access Control Method

4.1.1 Overview. The architectural model described in Section 1 is used in this section to provide a functional description of the Local Area Network CSMA/CD MAC sublayer.

The MAC 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 area network LLC sublayer. It is applicable to a general class of local area broadcast media suitable for use with the media access discipline known as Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

The LLC sublayer and the MAC 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 MAC sublayer. They are as follows:

(1) Data encapsulation (transmit and receive)
  (a) Framing (frame boundary delimitation, frame synchronization)
  (b) Addressing (handling of source and destination addresses)
  (c) Error detection (detection of physical medium transmission errors)
(2) Media Access Management
  (a) Medium allocation (collision avoidance)
  (b) Contention resolution (collision handling)

The remainder of this section provides a functional model of the CSMA/CD MAC method.

4.1.2 CSMA/CD Operation. This section provides 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 4.2 and 4.3. Specific implementations for CSMA/CD mechanisms that meet this standard are given in 4.4. Figure 4-1 provides the architectural model described functionally in the sections that follow.

The Physical Layer Signaling (PLS) component of the Physical Layer provides an interface to the MAC sublayer for the serial transmission of bits onto the physical media. For completeness, in the operational description that follows some of these functions are included as descriptive material. The concise specification of these functions is given in 4.2 for the MAC functions and in Section 7 for PLS.

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

4.1.2.1 Normal Operation

4.1.2.1.1 Transmission Without Contention. When a LLC sublayer requests the transmission of a frame, the Transmit Data Encapsulation component of the CSMA/CD MAC 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 MAC 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 4.2.3.3). 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 MAC 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 Signaling (PLS) component and deferring to passing traffic. When the medium is clear, frame transmission is initiated (after a brief inter-

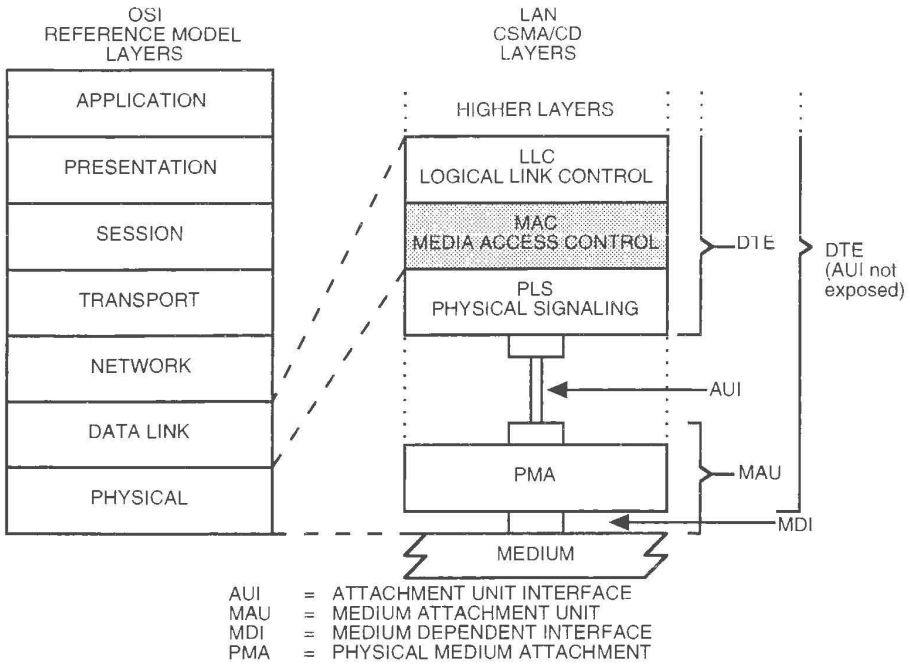


Fig 4-1
MAC Sublayer Partitioning, Relationship to the ISO Open Systems Interconnection (OSI) Reference Model

frame delay to provide recovery time for other CSMA/CD MAC sublayers and for the physical medium) The MAC 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 that 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. A functional description of the Physical Layer is given in Sections 7 and beyond.

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

4.1.2.1.2 Reception Without Contention. 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. The PLS passes subsequent bits up to the MAC sublayer, where 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 MAC sublayer, having observed carrier sense, has been 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 is removed, 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 (LLCDU) 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.

4.1.2.2 Access Interference and Recovery. If multiple stations attempt to transmit at the same time, it is possible for them to interfere with each other's transmissions, in spite of their attempts to avoid

this by deferring. When transmissions from two stations 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. Once the collision window has passed, a transmitting 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 (by way of 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 medium.

In the event of a collision, the transmitting station's Physical Layer initially notices the interference on the medium and then turns on the collision detect signal. This is noticed in turn by the Transmit Media Access Management component of the MAC sublayer, and collision handling begins. First, Transmit Media Access Management enforces the collision by transmitting a bit sequence called jam. In 4.4 an implementation that uses this enforcement procedure is provided. This ensures 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 another transmission attempt after a randomly selected time interval. Retransmission is attempted again 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 successive transmit 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 MAC sublayer's Receive Media Access Management component.

4.1.3 Relationships to LLC Sublayer and Physical Layer. The CSMA/CD MAC sublayer provides services to the LLC sublayer required for the transmission and reception of frames. Access to these services is specified in 4.3. The CSMA/CD MAC 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 (sub)layers.

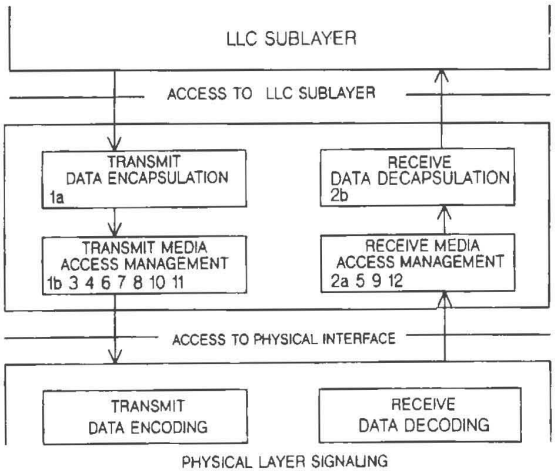
4.1.4 CSMA/CD Access Method Functional Capabilities. The following summary of the functional capabilities of the CSMA/CD MAC sublayer is intended as a quick reference guide to the capabilities of the standard, as depicted in Fig 4-2:

(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

  NOTE: Assumes 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) Discards or passes to Network Management all frames not addressed to the receiving station
(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 by way of FCS and verifies octet boundary alignment
(6) Delays transmission of frame bit stream for specified interframe gap period
(7) Halts transmission when collision is detected
(8) Schedules retransmission after a collision until a specified retry limit is reached
(9) Enforces collision to ensure propagation throughout network by sending jam message
(10) Discards received transmissions that are less than a minimum length
(11) Appends preamble, Start Frame Delimiter, DA, SA, length count, and FCS to all frames, and inserts pad field for frames whose LLC data length is less than a minimum value

(12) Removes preamble, Start Frame Delimiter, DA, SA, length count, FCS and pad field (if necessary) from received frames



NOTE: Numbers refer to functions listed in 4.1.4.

Fig 4-2
CSMA/CD Media Access Control Functions

4.2 CSMA/CD Media Access Control Method (MAC): Precise Specification

4.2.1 Introduction. A precise algorithmic definition is given in this section, providing procedural model for the CSMA/CD MAC process with a program in the computer language Pascal. See references [A2] and [A17] for resource material. Note whenever there is any apparent ambiguity concerning the definition of some aspect of the CSMA/CD MAC method, it is the Pascal procedural specification in 4.2.7 through 4.2.10 which should be consulted for the definitive statement. Sections 4.2.2 through 4.2.6 provide, in prose, a description of the access mechanism with the formal terminology to be used in the remaining subsections.

4.2.2 Overview of the Procedural Model. The functions of the CSMA/CD MAC method are presented below, modeled as a program written in the computer language Pascal. This procedural model is intended as the primary specification of the functions to be provided in any CSMA/CD MAC 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 shall 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 shall be considered.

4.2.2.1 Ground Rules for the Procedural Model

(1) First, it shall be emphasized that *the description of the MAC sublayer in a computer language is in no way intended to imply that procedures shall be implemented as a program executed by a computer.* The implementation may consist of any appropriate technology including hardware, firmware, software, or any combination.
(2) Similarly, it shall be emphasized that it is the behavior of any MAC sublayer implementations that shall 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.
(3) 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 MAC sublayer and are only serial-

ized for presentation to the Physical Layer. In reality, many implementations will instead handle frames serially on a bit, octet or word basis. This approach has not been reflected in the procedural model, since this only complicates the description of the functions without changing them in any way.

(4) The model consists of algorithms designed to be executed by a number of concurrent processes; these algorithms collectively implement the CSMA/CD procedure. The timing dependencies introduced by the need for concurrent activity are resolved in two ways:
   (a) *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 ReceiveFrame is always called well before the frame in question has started to arrive.
   (b) *Processes Versus Processes.* Among processes, no assumptions are made about relative speeds of execution. This means that each interaction between two processes shall 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 LLC and MAC procedures, although the actual parallel structure of the implementations is likely to vary.

4.2.2.2 Use of Pascal in the Procedural Model. Several observations need to be made regarding the method with which Pascal is used for the model. Some of these observations are as follows:

(1) Some limitations of the language have been circumvented to simplify the specification:
   (a) The elements of the program (variables and procedures, for example) are presented in logical groupings, in top-down order. Certain Pascal ordering restrictions have thus been circumvented to improve readability.
   (b) The *process* and *cycle* constructs of Concurrent Pascal, a Pascal derivative, 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.
   (c) The lack of variable array bounds in the language has been circumvented by treating frames as if they are always of a single fixed size (which is never actually specified). 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.
   (d) The use of a variant record to represent a frame (as fields and as bits) follows the spirit but not the letter of the Pascal Report, since it allows the underlying representation to be viewed as two different data types.
(2) The model makes no use of any explicit interprocess synchronization primitives. Instead, all interprocess interaction is done by way of 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 (microcode, hardware state-machines, etc.)

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

(1) Frame Transmitter Process
(2) Frame Receiver Process
(3) Bit Transmitter Process

(4) Bit Receiver Process
(5) Deference Process

This organization of the model is illustrated in Fig 4-3 and reflects the fact that the communication of entire frames is initiated by the client of the MAC sublayer; while the timing of collision backoff and of individual bit transfers is based on interactions between the MAC sublayer and the Physical-Layer-dependent bit time.

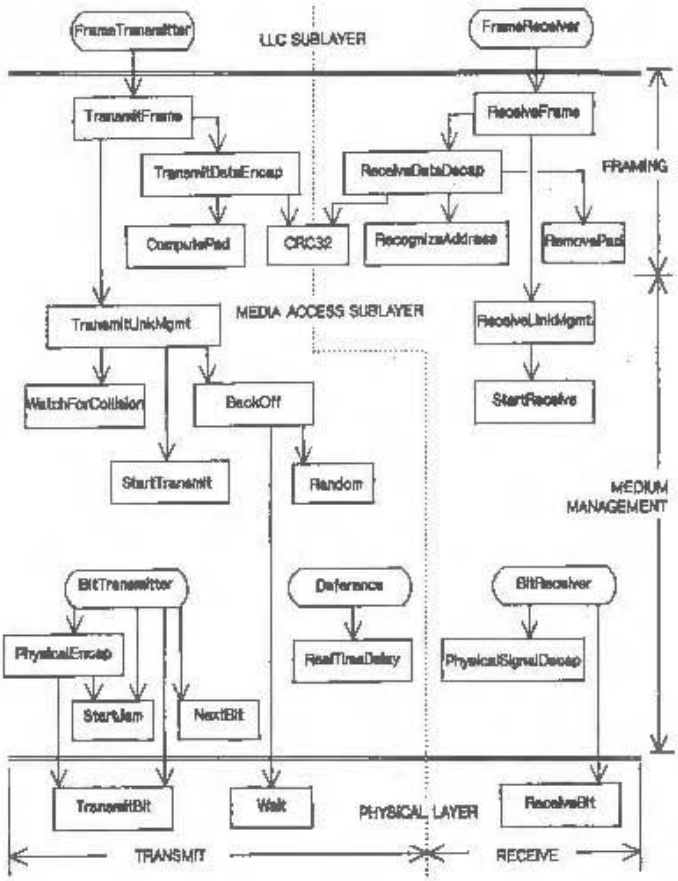
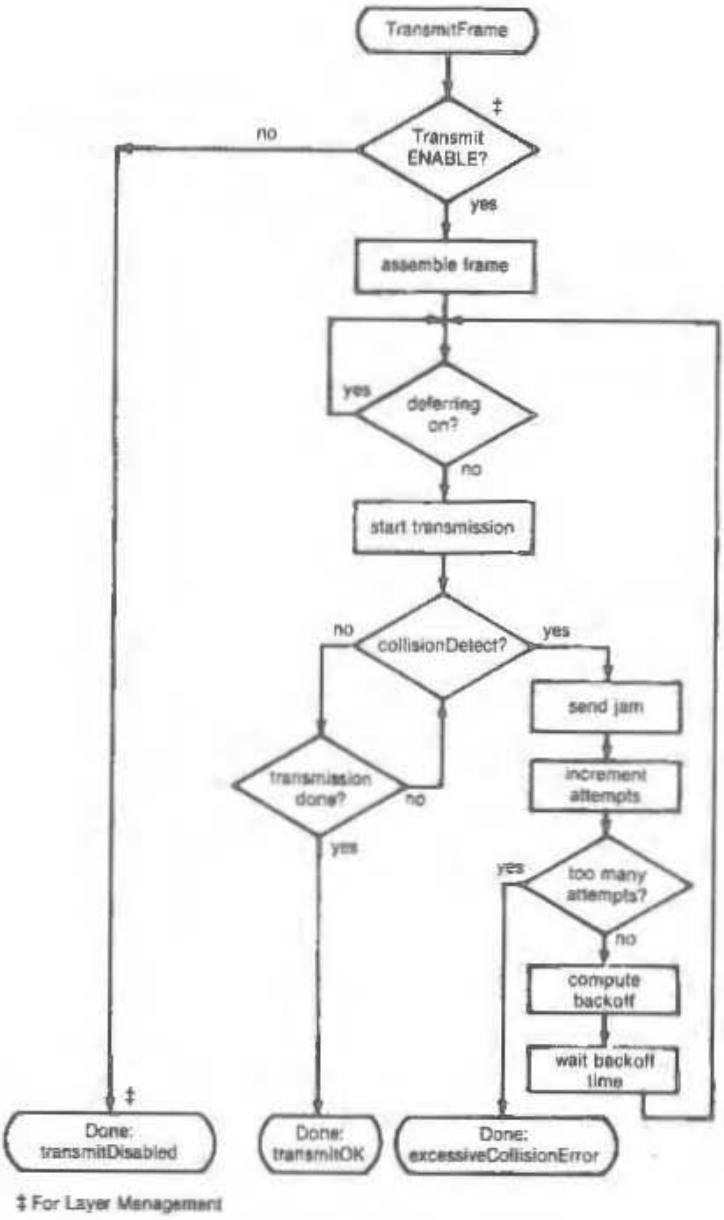


Fig 4-3
Relationship Among CSMA/CD Procedures

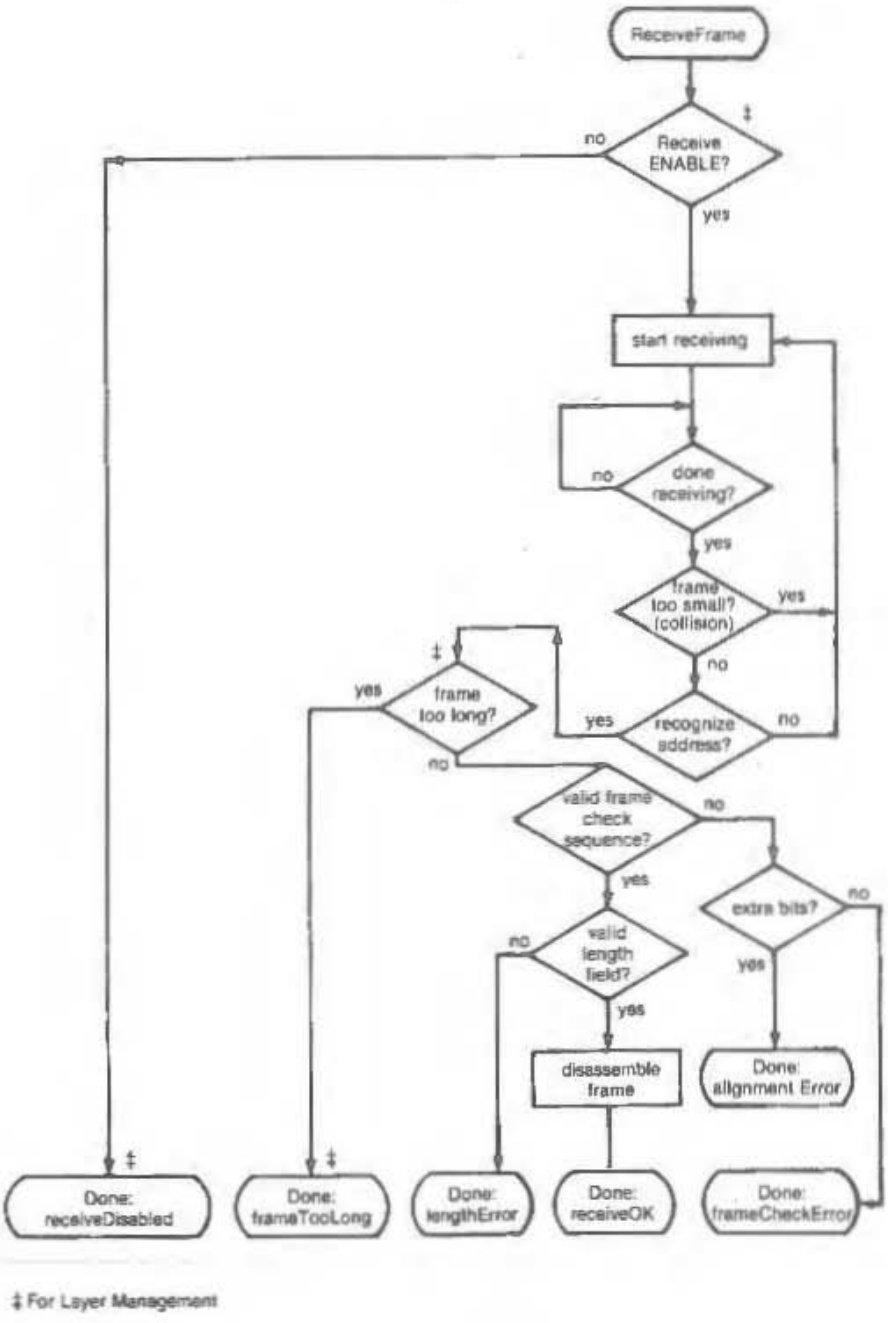
Figure 4-3 depicts the static structure of the procedural model, showing how the various processes and procedures interact by invoking each other. Figures 4-4 and 4-5 summarize the dynamic behavior of the model during transmission and reception, focusing on the steps that shall be performed, rather than the procedural structure that performs them. The usage of the shared state variables is not depicted in the figures, but is described in the comments and prose in the following sections.

4.2.2.4 Layer Management Extensions to Procedural Model. In order to incorporate network management functions, this Procedural Model has been expanded beyond that in ISO/IEC 8802-3 : 1990. Network management functions have been incorporated in two ways. First, 4.2.7–4.2.10, 4.3.2, and Fig 4-4 have been modified and expanded to provide management services. Second, Layer Management procedures have been added as 5.2.4. Note that Pascal variables are shared between Sections 4 and 5. Within the Pascal descriptions provided in Section 4, a "‡" in the left margin indicates a line that has been added to support management services. These lines are only required if Layer Management is being implemented.



(a) TransmitFrame

Fig 4-4
Control Flow Summary



(b) ReceiveFrame

Fig 4-4
Control Flow Summary

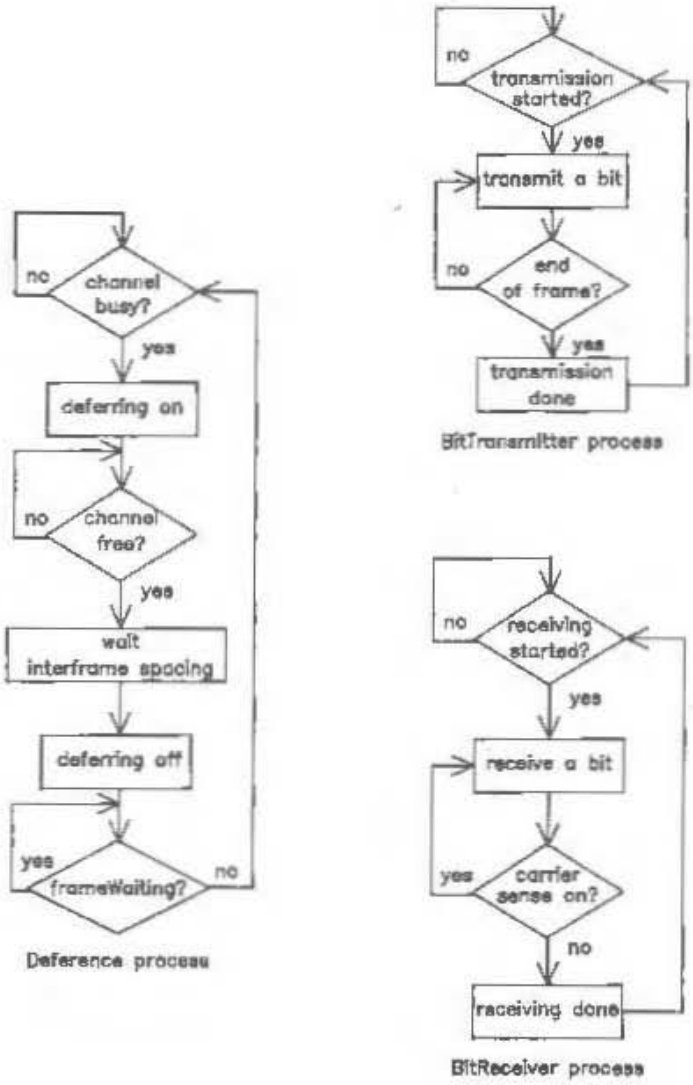


Fig 4-5
Control Flow: MAC Sublayer

These changes do not affect any aspect of the MAC behavior as observed at the LLC-MAC and MAC-PLS interfaces of ISO/IEC 8802-3 : 1990.

The Pascal procedural specification shall be consulted for the definitive statement when there is any apparent ambiguity concerning the definition of some aspect of the CSMA/CD MAC access method.

4.2.3 Frame Transmission Model. Frame transmission includes data encapsulation and Media Access management aspects:

(1) Transmit Data Encapsulation includes the assembly of the outgoing frame (from the values provided by the LLC sublayer) and frame check sequence generation.
(2) Transmit Media Access Management includes carrier deference, interframe spacing, collision detection and enforcement, and collision backoff and retransmission.

4.2.3.1 Transmit Data Encapsulation

4.2.3.1.1 Frame Assembly. The fields of the CSMA/CD MAC frame are set to the values provided by the LLC sublayer as arguments to the TransmitFrame operation (see 4.3) with the exception of the padding necessary to enforce the minimum framesize and the frame check sequence that is set to the CRC value generated by the MAC sublayer.

4.2.3.1.2 Frame Check Sequence Generation. The CRC value defined in 3.8 is generated and inserted in the frame check sequence field, following the fields supplied by the LLC sublayer.

4.2.3.2 Transmit Media Access Management

4.2.3.2.1 Carrier Deference. Even when it has nothing to transmit, the CSMA/CD MAC sublayer monitors the physical medium for traffic by watching the carrierSense signal provided by the PLS. Whenever the medium is busy, the CSMA/CD MAC sublayer defers to the passing frame by delaying any pending transmission of its own. After the last bit of the passing frame (that is, when carrierSense changes from true to false), the CSMA/CD MAC sublayer continues to defer for a proper interFrameSpacing (see 4.2.3.2.2).

If, at the end of the interFrameSpacing, a frame is waiting to be transmitted, transmission is initiated independent of the value of carrierSense. When transmission has completed (or immediately, if there was nothing to transmit) the CSMA/CD MAC 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.

NOTE: It is possible for the PLS carrier sense indication to fail to be asserted briefly during a collision on the media. If the Deference process simply times the interFrame gap based on this indication it is possible for a short interFrame gap to be generated, leading to a potential reception failure of a subsequent frame. To enhance system robustness the following optional measures, as specified in 4.2.8, are recommended when interFrame SpacingPart1 is other than zero:
(1) Upon completing a transmission, start timing the interpacket gap as soon as transmitting and carrierSense are both false.
(2) When timing an interFrame gap following reception, reset the interFrame gap timing if carrierSense becomes true during the first 2/3 of the interFrame gap timing interval. During the final 1/3 of the interval the timer shall not be reset to ensure fair access to the medium. An initial period shorter than 2/3 of the interval is permissible including zero.

4.2.3.2.2 Interframe Spacing. As defined in 4.2.3.2.1, the rules for deferring to passing frames ensure 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 4.4.

4.2.3.2.3 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:

(1) It is an upper bound on the acquisition time of the medium.
(2) It is an upper bound on the length of a frame fragment generated by a collision.
(3) It is the scheduling quantum for retransmission.

To fulfill all three functions, the slot time shall 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 4.4.

4.2.3.2.4 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 Media Access implementation, however, the implementation shall not be intentionally designed to be the 32-bit CRC value corresponding to the (partial) frame transmitted prior to the jam.

4.2.3.2.5 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 integer 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:

$$0 \le r < 2^k$$

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 that will 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.

4.2.3.3 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 MAC sublayer shall append extra bits in units of octets, after the end of the LLC data field but prior to calculating, and appending, the FCS. The number of extra bits shall be sufficient to ensure that the frame, from the DA field through the FCS field inclusive, is at least minFrameSize bits. The content of the pad is unspecified.

4.2.4 Frame Reception Model. CSMA/CD MAC sublayer frame reception includes both data decapsulation and Media Access management aspects:

(1) 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.
(2) Receive Media Access Management comprises recognition of collision fragments from incoming frames and truncation of frames to octet boundaries.

4.2.4.1 Receive Data Decapsulation

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

(1) *Individual Addresses.* The CSMA/CD MAC sublayer recognizes and accepts any frame whose DA field contains the individual address of the station.
(2) *Group Addresses.* The CSMA/CD MAC sublayer recognizes and accepts any frame whose DA field contains the Broadcast address.

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

4.2.4.1.2 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.

4.2.4.1.3 Frame Disassembly. Upon recognition of the Start Frame Delimiter at the end of the preamble sequence, the CSMA/CD MAC sublayer accepts the frame. If there are no errors, the frame is disassembled and the fields are passed to the LLC sublayer by way of the output parameters of the ReceiveFrame operation.

4.2.4.2 Receive Media Access Management

4.2.4.2.1 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, that indicate ill-framed data: the frame may be too long, or its length may not be an integer number of octets.

(1) *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.
(2) *Integer Number of Octets in Frame.* Since the format of a valid frame specifies an integer number of octets, only a collision or an error can produce a frame with a length that is not an integer multiple of 8 bits. Complete frames (that is, not rejected as collision fragments; see 4.2.4.2.2) that do not contain an integer 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.

4.2.4.2.2 Collision Filtering. The smallest valid frame shall 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.

4.2.5 Preamble Generation. In a LAN implementation, most of the Physical Layer components are allowed to provide valid output some number of bit times after being presented valid input signals. 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 TransmitLinkMgmt to transmit the first bit of a new frame, PhysicalSignalEncap shall first transmit the preamble, a bit sequence used for physical medium stabilization and synchronization, followed by the Start Frame Delimiter. If, while transmitting the preamble, the PLS asserts the collision detect signal, any remaining preamble bits shall 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, for Manchester encoding, it appears as a periodic waveform on the medium that enables bit synchronization. It should be noted that the preamble ends with a "0."

4.2.6 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.

4.2.7 Global Declarations. This section provides detailed formal specifications for the CSMA/CD MAC sublayer. It is a specification of generic features and parameters to be used in systems implementing this media access method. Subsection 4.4 provides values for these sets of parameters for recommended implementations of this media access mechanism.

4.2.7.1 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:

```
const
   addressSize = ... ; {16 or 48 bits in compliance with 3.2.3}
   lengthSize = 16; {in bits}
   LLCdataSize = ...; {LLC Data, see 4.2.2.2, (1)(c)}
   padSize = ...; {in bits, = max (0, minFrameSize – (2 × addressSize + lengthSize + LLCdataSize +
                 crcSize))}.
   dataSize = ...; {= LLCdataSize + padSize}
   crcSize = 32; {32 bit CRC = 4 octets}
   frameSize = ...; {= 2 × addressSize + lengthSize + dataSize + crcSize, see 4.2.2.2(1)}
   minFrameSize = ... ; {in bits, implementation-dependent, see 4.4}
   slotTime = ... ; {unit of time for collision handling, implementation-dependent, see 4.4}
   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}
```

4.2.7.2 Transmit State Variables. The following items are specific to frame transmission. (See also 4.4.)

```
const
  interFrameSpacing = ... ; {minimum time between frames}
  interFrameSpacingPart1= ...;{duration of first portion of interFrame timing. In range 0 up to 2/3
            interFrameSpacing}
  interFrameSpacingPart2= ...;{duration of remainder of interFrame timing. Equal to
            interFrameSpacing – interFrameSpacingPart1}
  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 outgoingFrame}
  newCollision: Boolean; {Indicates that a collision has occurred but has not yet been jammed}
  transmitSucceeding: Boolean; {Running indicator of whether transmission is succeeding}
```

4.2.7.3 Receive State Variables. The following items are specific to frame reception. (See also 4.4.)

```
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 succeeding}
  validLength: Boolean; {Indicator of whether received frame has a length error}
  exceedsMaxLength: Boolean; {Indicator of whether received frame has a length longer than the
            maximum permitted length}
```

4.2.7.4 Summary of Interlayer Interfaces

(1) The interface to the LLC sublayer, defined in 4.3.2, is summarized below:

```
type
‡   TransmitStatus = (transmitDisabled, transmitOK, excessiveCollisionError);
                {Result of TransmitFrame operation}
‡   ReceiveStatus = (receiveDisabled, receiveOK, frameTooLong, frameCheckError, lengthError,
                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}
```

(2) The interface to the Physical Layer, defined in 4.3.3, is summarized below:

```
var
   carrierSense: Boolean; {Indicates incoming bits}
   transmitting: Boolean; {Indicates outgoing bits}
   wasTransmitting: Boolean; {Indicates transmission in progress or just completed}
   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}
```

4.2.7.5 State Variable Initialization. The procedure Initialize must be run when the MAC 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.

If Layer Management is implemented, the Initialize procedure shall only be called as the result of the initializeMAC action (5.2.2.2.1).

```
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}
```

4.2.8 Frame Transmission. The algorithms in this section define MAC 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
   if transmitEnabled then
   begin
      TransmitDataEncap;
      TransmitFrame := TransmitLinkMgmt
   end
   else TransmitFrame := transmitDisabled
end; {TransmitFrame}
```

If transmission is enabled, 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;
    lateCollisionCount := 0;
    deferred := false; {initialize}
    excessDefer := false;
    while(attempts < attemptLimit) and (not transmitSucceeding)do
    begin {loop}
      if attempts > 0 then BackOff;
      frameWaiting := true;
      lateCollisionError := false;
‡     while deferring do {defer to passing frame, if any}
      begin
‡       nothing;
        deferred := true;
      end;
      frameWaiting := false;
      StartTransmit;
      while transmitting do WatchForCollision;
      if lateCollisionError then lateCollisionCount := lateCollisionCount + 1;
      attempts := attempts+1
    end; {loop}
    if transmitSucceeding then TransmitLinkMgmt := transmitOK
    else TransmitLinkMgmt := excessiveCollisionError;
    LayerMgmtTransmitCounters; {update transmit and transmit error counters in 5.2.4.2}
    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
      if currentTransmitBit > (minFrameSize – headerSize) then
         lateCollisionError := true;
      newCollision := true;
      transmitSucceeding := false
   end
end; {WatchForCollision}
```

WatchForCollision, upon detecting a collision, updates newCollision to ensure proper jamming by the BitTransmitter process. The current transmit bit number is checked to see if this is a late collision. If the collision occurs later than a collision window of 512 bit times into the packet, it is considered as evidence of a late collision. The point at which the collision is received is determined by the network media propagation time and the delay time through a station and, as such, is implementation-dependent (see 4.1.2.2). An implementation may optionally elect to end retransmission attempts after a late collision is detected.

After transmission of the jam has been 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 × 2;
   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}
            wasTransmitting := transmitting;
            while carrierSense or transmitting then
               wasTransmitting := wasTransmitting or transmitting;
            if wasTransmitting do
               begin
                  StartRealTimeDelay; {time out first part interframe gap}
                  while RealTimeDelay(interFrameSpacingPart1) do nothing
               end
            else
               begin
                  StartRealTimeDelay;
                  repeat
                     while carrierSense do StartRealTimeDelay
                  until not RealTimeDelay(interFrameSpacingPart1)
               end;
            StartRealTimeDelay; {time out second part interframe gap}
            while RealTimeDelay(interFrameSpacingPart2) do nothing;
            deferring := false; {allow new transmissions to proceed}
            while frameWaiting do nothing; {allow waiting transmission if any}
         end {main loop}
      end; {Deference}

      procedure StartRealTimeDelay
         begin
            {reset the realtime timer and start it timing}
         end; {StartRealTimeDelay}

      function RealTimeDelay (μsec:real): Boolean;
         begin
            {return the value true if the specified number of microseconds have
            not elapsed since the most recent invocation of StartRealTimeDelay,
            otherwise return the value false}
         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}
      procedure 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.)

4.2.9 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
      if receiveEnabled then
      repeat
        ReceiveLinkMgmt;
        ReceiveFrame := ReceiveDataDecap;
      until receiveSucceeding
      else
        ReceiveFrame := receiveDisabled
    ; {ReceiveFrame}
```

If enabled, 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;
‡     var status: ReceiveStatus; {holds receive status information}
    begin
‡     with incomingFrame do
‡     begin
‡       view := fields;
        receiveSucceeding := RecognizeAddress (incomingFrame, destinationField);
```

```
        receiveSucceeding := LayerMgmtRecognizeAddress (destinationField);
‡       if receiveSucceeding then
        begin {disassemble frame}
          destinationParam := destinationField;
          sourceParam := sourceField;
          lengthParam := lengthField;
          dataParam := RemovePad (lengthField, dataField);
          exceedsMaxLength := ...; {check to determine if receive frame size exceeds the maximum
                                    permitted frame size (maxFrameSize)}
          if exceedsMaxLength then status := frameTooLong;
          else
          if fcsField = CRC32 (incomingFrame) then
          begin
‡           if validLength then status := receiveOK
‡           else status := lengthError
          end
          else
          begin
‡           if excessBits = 0 then status := frameCheckError
‡           else status := alignmentError;
          end;
          LayerMgmtReceiveCounters(status);
            {update receive and receive error counters in 5.2.4.3}
          view := bits
        end {disassemble frame}
‡     end {with incomingFrame}
‡     ReceiveDataDecap := status;
    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
        validLength := {Check to determine if value represented by lengthParam matches received
                        LLCdataSize};
        if validLength then
          RemovePad := {truncate the dataParam (when present) to value represented by lengthParam
                        (in octets) and return the result}
          else
          RemovePad := dataParam
    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}
        excessBits := frameSize mod 8;
```

```
        frameSize := frameSize – excessBits; {truncate to octet boundary}
        receiveSucceeding :– (frameSize ≥ minFrameSize); {reject collision fragments}
    until receiveSucceeding
end; {ReceiveLinkMgmt}

procedure StartReceive;
begin
    currentReceiveBit := 1;
    receiving := true
end; {StartReceive}
```

The BitReceiver process runs asynchronously, receiving bits from the medium at the 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; {BitReceiver}

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

4.2.10 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 (that is, while waiting for some event) is cast as repeated calls on this procedure.

4.3 Interfaces to/from Adjacent Layers

4.3.1 Overview. The purpose of this section is to provide precise definitions of the interfaces between the architectural layers defined in Section 1 in compliance with the Media Access Service Specification given in Section 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 precise MAC sublayer specification (see 4.2). Each interface is described as a set of procedures or shared variables, or both, 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 4.2, 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 MAC sublayers, and need not be reflected in the implementation interface between a station and its sublayer.

4.3.2 Services Provided by the MAC Sublayer. The services provided to the LLC sublayer by the MAC sublayer are transmission and reception of LLC frames. The interface through which the LLC sublayer uses the facilities of the MAC 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. Note that the service_class defined in 2.3.1 is ignored by CSMA/CD MAC.

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

The transmitDisabled status code indicates that the transmitter is not enabled. Successful transmission is indicated by the status code transmittOK; the code excessiveCollisionError indicates that the transmission attempt was aborted due to the 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 with a status code:

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

‡ *type* Receive Status = (receiveDisabled, receive OK, frameTooLong, frameCheck Error, length Error, alignmentError);

The receiveDisabled status indicates that the receiver is not enabled. Successful reception is indicated by the status code receiveOK. The frameTooLong error indicates that a frame was received whose frameSize was beyond the maximum allowable frame size. The code frameCheckError indicates that the frame received was damaged by a transmission error. The lengthError indicates the lengthParam value was inconsistent with the frameSize of the received frame. The code alignmentError indicates that the frame received was damaged, and that in addition, its length was not an integer number of octets.

4.3.3 Services Required from the Physical Layer. The interface through which the CSMA/CD MAC sublayer uses the facilities of the Physical Layer consists of a function, a pair of procedures and three Boolean variables.

Function:	*Procedures*:	*Variables*:
ReceiveBit	TransmitBit	collisionDetect
	Wait	carrierSense
		transmitting

During transmission, the contents of an outgoing frame are passed from the MAC sublayer to the Physical Layer by way of 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 signaled to the Physical Layer by way of the variable transmitting:

var transmitting: Boolean;

Before sending the first bit of a frame, the MAC 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 MAC sublayer sets transmitting to false to indicate the end of the frame.

The presence of a collision in the physical medium is signaled to the MAC 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 MAC sublayer shall handle this possibility by monitoring collisionDetect concurrently with its transmission of outgoing bits. See 4.2 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 MAC 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 MAC sublayer shall immediately request the next bit until all bits of the frame have been received. (See 4.2 for details.)

The overall event of data being received is signaled to the MAC sublayer by the variable carrierSense:

var carrierSense: Boolean;

When the Physical Layer sets carrierSense to true, the MAC sublayer shall immediately begin retrieving the incoming bits by the ReceiveBit operation. When carrierSense subsequently becomes false, the MAC 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 MAC sublayer. (See 4.2 for details.)

The MAC sublayer shall 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 MAC 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 MAC 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 4.4 for a selection of physical media.

4.4 Specific Implementations

4.4.1 Compatibility Overview. To provide total compatibility at all levels of the standard, it is required that each network component implementing the CSMA/CD MAC sublayer procedure adheres rigidly to these specifications. The information provided in 4.4.2.1 below provides design parameters for a specific implementation of this access method. Variations from these values result in a system implementation that violates the standard.

4.4.2 Allowable Implementations

4.4.2.1 Parameterized Values. The following table identifies the parameter values that shall be used in the 10 Mb/s implementation of a CSMA/CD MAC procedure. The primary assumptions are that the physical medium is a baseband coaxial cable with properties given in the Physical Layer section(s) of this standard.

Parameters	Values
slotTime	512 bit times
interFrameGap	9.6 μs
attemptLimit	16
backoffLimit	10
jamSize	32 bits
maxFrameSize	1518 octets
minFrameSize	512 bits (64 octets)
addressSize	48 bits

WARNING: Any deviation from the above plans specified for a 10 Mb/s system may affect proper operation of the LAN.

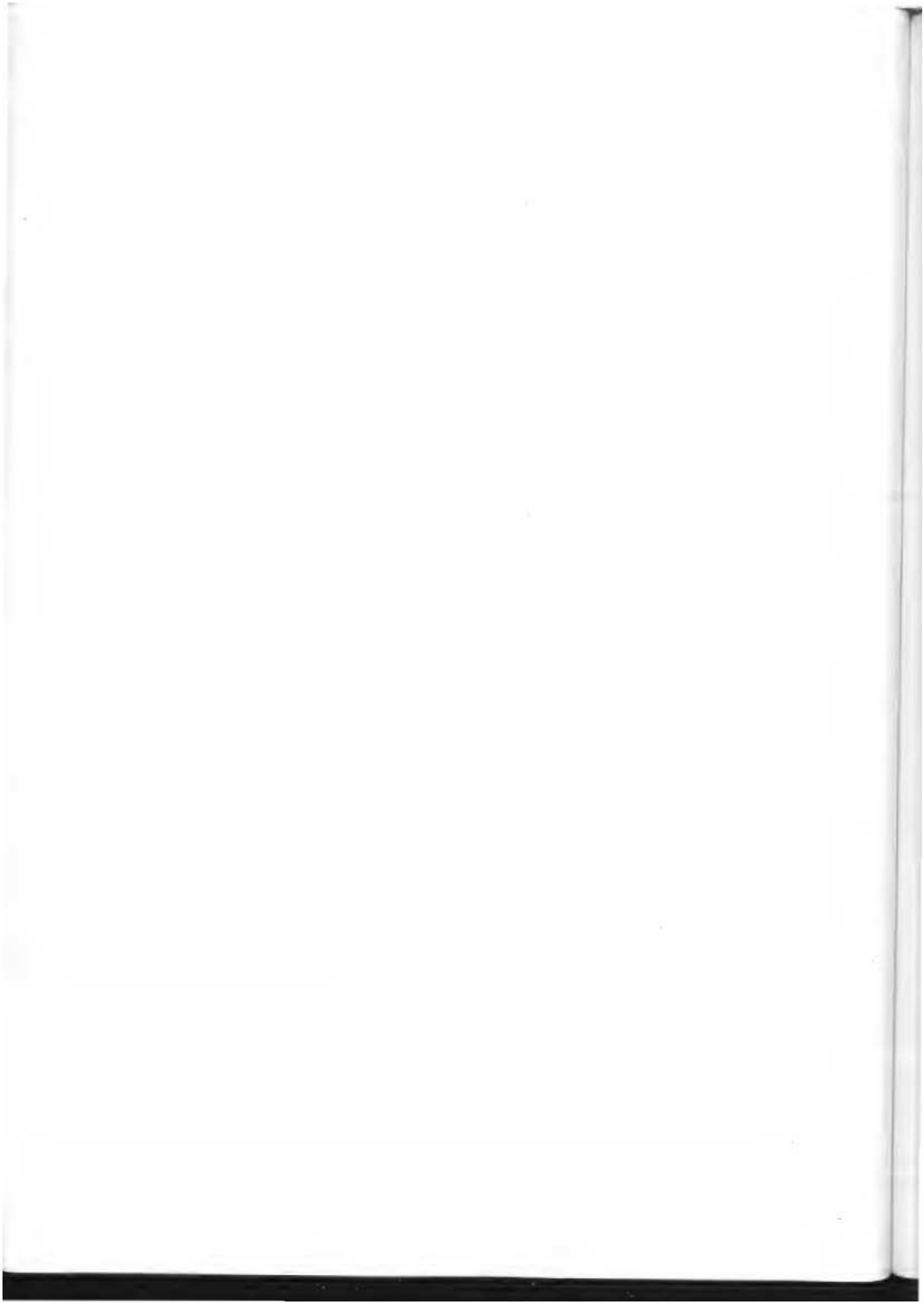
See also DTE Deference Delay in 12.9.2.

4.4.2.2 Parameterized Values. The following parameter values shall be used for 1BASE5 implementations:

Parameters	Values
slotTime	512 bit times
interFrameGap	96 μs
attemptLimit	16
backoffLimit	10
jamSize	32 bits
maxFrameSize	1518 octets
minFrameSize	512 bits (64 octets)
addressSize	48 bits

See also DTE Deference Delay in 12.9.2.

WARNING: Any deviation from the above specified values may affect proper operation of the network.



5. Layer Management

5.1 Introduction. This section provides the Layer Management specification for networks based on the CSMA/CD access method. It defines facilities comprised of a set of statistics and actions needed to provide Layer Management services. The information in this chapter should be used in conjunction with the Procedural Model defined in 4.2.7–4.2.10. The Procedural Model provides a formal description of the relationship between the CSMA/CD Layer Entities and the Layer Management facilities.

This Layer Management specification has been developed in accordance with the OSI management architecture as specified in the ISO Management Framework document, ISO/IEC 7498-4:1989 [20]. It is independent of any particular management application or management protocol.

The management facilities defined in this standard may be accessed both locally and remotely. Thus, the Layer Management specification provides facilities that can be accessed from within a station or can be accessed remotely by means of a peer management protocol operating between application entities.

In CSMA/CD no peer management facilities are necessary for initiating or terminating normal protocol operations or for handling abnormal protocol conditions. The monitoring of these activities is done by the carrier sense and collision detection mechanisms. Since these activities are necessary for normal operation of the protocol, they are not considered to be a function of Layer Management and are therefore not discussed in this section.

At this time, this standard does not include management facilities that address the unique features of repeaters or of 10BROAD36 broadband MAUs.

5.1.1 Systems Management Overview. Within the ISO Open Systems Interconnection (OSI) architecture, the need to handle the special problems of initializing, terminating, and monitoring on-going activities and assisting in their harmonious operations, as well as handling abnormal conditions, is recognized. These needs are collectively addressed by the systems management component of the OSI architecture.

The systems management component may conceptually be subdivided into a System Management Application Entity (SMAE) and Layer Management Entities (LMEs). In addition, a Management Protocol is required for the exchange of information between systems on a network. This Layer Management standard is independent of any particular Management Protocol.

The SMAE is concerned with the management of resources and their status across all layers of the OSI architecture. The System Management Application facilities have been grouped into five entities: Configuration, Fault, Performance, Security, and Accounting.

Configuration and Name Management is concerned with the initialization, normal operation, and closedown of communication facilities. It is also concerned with the naming of these resources and their interrelationship as part of a communication system. Fault Management is concerned with detection, isolation, and correction of abnormal operations. Performance Management is concerned with evaluating the behavior and the effectiveness of the communication activities. Security Management is concerned with monitoring the integrity and controlling access to the communication facilities. Accounting Management is concerned with enabling charges to be established and cost to be assigned and providing information on tariffs for the use of communication resources.

This Layer Management standard, in conjunction with the Layer Management standards of other layers, provides the means for the SMAE to perform its various functions. Layer Management collects information needed by the SMAE from the MAC and Physical Layers. It also provides a means for the SMAE to exercise control over those layers. This Layer Management standard is independent of any specific SMAE.

The SMAE has a conceptual interface to an LME concerned with the actual monitoring and control of a specific layer. The LME interfaces directly only with the SMAE, to whom it provides Layer Management facilities.

Strictly, only those management activities that imply actual exchanges of information between peer entities are pertinent to OSI architecture. Therefore, only the protocols needed to conduct such exchanges are candidates for standardization. As a practical matter, however, the specification of the Layer Management facilities provided across the conceptual Layer Management Interface (LMI) between the LME and SMAE is required. Standardization of these facilities will make practical the use of higher layer protocols for the control and maintenance of LANs.

There are two interfaces that relate the various management entities. These are as follows:

(1) The conceptual Layer Management Interface between the SMAE and LME.

(2) The normal layer service interface for peer-to-peer communication.

The relationship between the various management entities and the layer entities according to the ISO Model is shown in Fig 5-1.

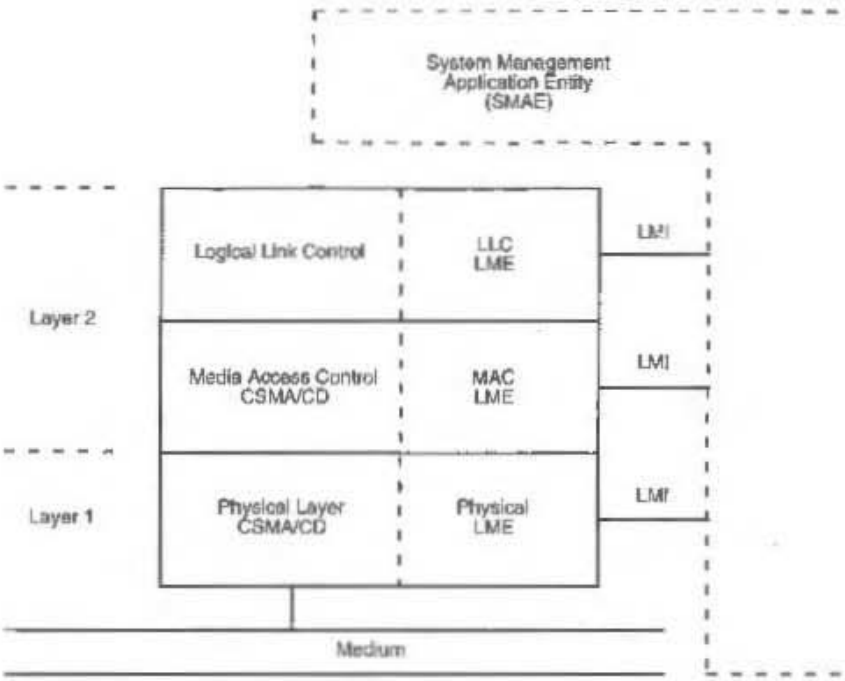


Fig 5-1
Relationship Between the Various Management Entities and Layer Entities According to the ISO Open Systems Interconnection (OSI) Reference Model

The conceptual LMI between the SMAE and the LME will be described in this standard in terms of the Layer Management facilities provided. It is particularly important that these facilities be defined because they may be indirectly requested on behalf of a remote SMAE. The use of this specification by other management mechanisms is not precluded.

5.1.2 Layer Management Model. The Layer Management facilities provided by the CSMA/CD MAC and Physical Layer LMEs, using the conceptual LMI, enable the SMAE to manipulate management counters and initiate actions within the layers. The LMI provides a means to monitor and control the facilities of the LMEs.

The CSMA/CD MAC/Physical Layer LMEs, in order to support the above facilities, offer a set of statistics and actions that constitute the conceptual LMI. The client of the LME (i.e., the SMAE) is thus able to read these statistics and to execute actions.

It is by executing these actions that the SMAE can cause certain desired effects on the MAC or Physical Layer Entities. The precise semantics of the relationship between the CSMA/CD Layer Entities and the Layer Management facilities are defined in 4.2.7–4.2.10 and in 5.2.4.

5.2 Management Facilities

5.2.1 Introduction. This section of the standard defines the Layer Management facilities for the IEEE 802.3 CSMA/CD MAC and Physical Layers. The intent of this standard is to furnish a management specification that can be used by the wide variety of different devices that may be attached to a network specified by ISO/IEC 8802-3. Thus, a comprehensive list of management facilities is provided.

The improper use of some of the facilities described in this section may cause serious disruption of the network. It should be noted that access to these facilities can only be obtained by means of the SMAE. To avoid duplication by each LME, and in accordance with ISO management architecture, any necessary security provisions should be provided by the SMAE. This can be in the form of specific SMAE security features or in the form of security features provided by the peer-to-peer communication facilities used by the SMAE

The statistics and actions are categorized into the three classifications defined as follows:

Mandatory—Shall be implemented.
Recommended—Should be implemented if possible.
Optional—May be implemented.

All counters defined in this specification are wraparound counters. Wraparound counters are those that automatically go from their maximum value (or final value) to zero and continue to operate. These unsigned counters do not provide for any explicit means to return them to their minimum (zero), i.e., reset. Because of their nature, wraparound counters should be read frequently enough to avoid loss of information.

5.2.2 MAC Sublayer Management Facilities. This section of the standard defines the Layer Management facilities specific to the MAC sublayer.

5.2.2.1 MAC Statistics. The statistics defined in this section are implemented by means of counters.

In the following definitions, the term "Read only" specifies that the object cannot be written to by the client of the LME.

Frame fragments are not included in any of the statistics in this section unless otherwise stated.

The Layer Management Model in 5.2.4 and the Pascal Procedural Model in 4.2.7–4.2.10 defines the semantics of these statistics in terms of the behavior of the MAC sublayer.

5.2.2.1.1 MAC Transmit Statistics Descriptions

(1) Number of framesTransmittedOK: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are successfully transmitted. This counter is incremented when the TransmitStatus is reported as transmitOK. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(2) Number of singleCollisionFrames: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are involved in a single collision and are subsequently transmitted successfully. This counter is incremented when the result of a transmission is reported as transmitOK and the attempt value is 2. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(3) Number of multipleCollisionFrames: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are involved in more than one collision and are subsequently transmitted successfully. This counter is incremented when the TransmitStatus is reported as transmitOK and the value of the attempts variable is greater than 2 and less than or equal to attemptLimit. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(4) Number of collisionFrames: Recommended, Read only, Array [1..attemptLimit – 1] of 32 bit counters.
This array provides a histogram of collision activity. The indices of this array (1 to attemptLimit — 1) denote the number of collisions experienced in transmitting a frame. Each element of this array contains a counter that denotes the number of frames that have experienced a specific number of collisions. When the TransmitStatus is reported as transmitOK and the value of the attempts variable equals n, then collisionFrames[n–1] counter is incremented. The elements of this array are incremented in the LayerMgmtTransmitCounters procedure (5.2.4.2).