Ethernet

The Ethernet

A Local Area Network

Data Link Layer
and
Physical Layer
Specifications

digital

Digital Equipment Corporation
Maynard, MA

intel

Intel Corporation
Santa Clara, CA

XEROX

Xerox Corporation
Stamford, CT

Version 2.0

November, 1982

AA-K759B-TK

IMPORTANT INFORMATION AND DISCLAIMERS

1. This specification includes subject matter relating to a patent(s) of Xerox Corporation. No license under such patent(s) is granted by implication, estoppel or otherwise as a result of publication of this specification. Applicable licenses may be obtained from Xerox Corporation.

2. This specification is furnished for informational purposes only. Digital, Intel, and Xerox do not warrant or represent that this specification or any products made in conformance with it will work in the intended manner or be compatible with other products in a network system. Nor do they assume responsibility for any errors that the specification may contain, or have any liabilities or obligations for damages (including but not limited to special, indirect or consequential damages) arising out of or in connection with the use of this specification in any way. Digital, Intel and Xerox products may follow or deviate from the specification without notice at any time.

3. No representations or warranties are made that this specification or anything made from it is or will be free from infringements or patents of third persons.

Preface

This document contains the specification of the Ethernet, a local area network developed jointly by Digital Equipment Corporation, Intel Corporation, and Xerox Corporation. The Ethernet specification is the result of an extensive collaborative effort of the three corporations, and several years of work at Xerox on an earlier prototype Ethernet.

This specification is intended as a design reference document, rather than an introduction or tutorial. Readers seeking introductory material are directed to the reference list in Section 2, which cites several papers describing the intent, theory, and history of the Ethernet.

This document contains 8 sections, falling into four main groups:

Sections 1, 2, and 3 provide an overall description of the Ethernet, including its goals, and the scope of the specification.

Sections 4 and 5 describe the architectural structure of the Ethernet in terms of a functional model consisting of two layers, the Data Link Layer and the Physical Layer.

Sections 6 and 7 specify the two layers in detail, providing the primary technical specification of the Ethernet.

Section 8 provides a description and specification for a configuration testing protocol for Network Management services. This protocol provides a minimum capability for testing any station's ability to communicate with other stations on the network.

Readers wishing to obtain an initial grasp of the organization and content of the specification will be best served by reading Sections 1, 3, and 4. Readers involved in actual implementation of the Ethernet will find Sections 5, 6, 7, and 8 to contain the central material of the specification. Section 2 provides references, and the appendices provide supplementary material.

The approach taken in the specification of the Data Link Layer in Section 6 is a procedural one; in addition to describing the necessary algorithms in English and control flow charts, the specification presents these algorithms in the language Pascal. This approach makes clear the required behavior of the Data Link Layer, while leaving individual implementations free to exploit any appropriate technology.

Because the procedural approach is not suitable for specifying the details of the Physical Layer, Section 7 uses carefully worded English prose and numerous figures and tables to specify the necessary parameters of this layer.

Some aspects of the Ethernet are necessarily discussed in more than one place in this specification. Whenever any doubt arises concerning the official definition in such a case, the reader should utilize the Pascal procedural specification of the Data Link Layer in Section 6.5, and the detailed prose specification of the Physical Layer in Sections 7.2 through 7.9.

One aspect of an overall network architecture which is addressed by this specification is network management. The network management facility performs operation, maintenance, and planning functions for the network:

- Operation functions include parameter setting, such as address selection.
- Maintenance functions provide for fault detection, isolation, and repair.
- Planning functions include collection of statistical and usage information, necessary for planned network growth.

While network management itself is properly performed outside the Ethernet Data Link and Physical Layers, it requires appropriate additional interfaces to those layers. A functional description of the network management interfaces is given in Section 4.5, while detailed procedural models are given in Sections 5 and 6. Configuration control and overall network management for Ethernets which may interconnect machines from different manufacturers, implementing different, perhaps incompatible communication software at the client layer of Ethernet, will require a minimum, common communication capability for network management purposes. The specification for the configuration testing protocol appears in Section 8.

Version 2.0 of the Ethernet specification reflects the experience of the three corporations in designing equipment to the Version 1.0 specification. Version 2.0 includes network management functions and better defines the details of the physical channel signalling. Version 2.0 is upward compatible with Version 1.0. Equipment designed to the two specifications is interoperable.

Section 7.6.4 on Repeaters is incomplete. Further study of the repeater design is desirable, with the goal to relax some of the configuration restrictions in Section 7.1.5.

Version 2.0 of the Ethernet specification is substantially compatible with standards for CSMA/CD local area networks being developed by IEEE and ECMA. The three corporations have been active participants in these standard efforts and have now endorsed their documents. Version 2.0 of the Ethernet specification is an interim document. We expect that future work will take place in these standards bodies.

Table of Contents

Preface 3

1. INTRODUCTION 1

2. REFERENCES 3

3. GOALS AND NON-GOALS 5

3.1 Goals 5

3.2 Non-Goals. 6

4. FUNCTIONAL MODEL OF THE ETHERNET ARCHITECTURE 7

4.1 Layering 9

4.2 Data Link Layer 10

4.3 Physical Layer 12

4.4 Network Management 12

4.5 Ethernet Operation and the Functional Model 15

4.5.1 Transmission Without Contention 15

4.5.2 Reception Without Contention. 15

4.5.3 Collisions: Handling of Contention 16

4.5.4 Functional Description of Network Management Interface. 17

5. INTER-LAYER INTERFACES 19

5.1 Client Layer to Data Link Layer 19

5.2 Data Link Layer to Physical Layer 20

5.3 Data Link to Network Management Interface 23

6. ETHERNET DATA LINK LAYER SPECIFICATION 26

6.1 Data Link Layer Overview and Model 26

6.2 Frame Format 26

6.2.1 Address Fields. 28

6.2.1.1 Destination Address Field. 29

6.2.1.2 Source Address Field 29

6.2.2 Type Field 29

6.2.3 Data Field 29

6.2.4 Frame Check Sequence Field. 29

6.2.5 Frame Size Limitations 30

6.3 Frame Transmission 30

6.3.1 Transmit Data Encapsulation 31

6.3.1.1 Frame Assembly 31

6.3.1.2 Frame Check Sequence Generation. 31

6.3.2 Transmit Link Management 31
6.3.2.1 Carrier Deference 31
6.3.2.2 Interframe Spacing 31
6.3.2.3 Collision Handling 32
6.3.2.3.1 Collision Detection and Enforcement . 32
6.3.2.3.2 Collision Backoff and Retransmission . 32

6.4 Frame Reception 33
6.4.1 Receive Data Decapsulation 33
6.4.1.1 Framing 33
6.4.1.1.1 Maximum Frame Size 33
6.4.1.1.2 Integral Number of Octets in Frame . 33
6.4.1.2 Address Recognition 34
6.4.1.2.1 Physical Addresses 34
6.4.1.2.2 Multicast Addresses 34
6.4.1.3 Frame Check Sequence Validation 34
6.4.1.4 Frame Disassembly 34
6.4.2 Receive Link Management 34
6.4.2.1 Collision Filtering 35

6.5 The Data Link Layer Procedural Model 35
6.5.1 Overview of the Procedural Model 35
6.5.1.1 Ground Rules for the Procedural Model . . . 35
6.5.1.2 Use of Pascal in the Procedural Model 36
6.5.2 Procedural Model 37
6.5.2.1 Global Declarations 43
6.5.2.1.1 Common Constants and Types . . . 43
6.5.2.1.2 Transmit State Variables 43
6.5.2.1.3 Receive State Variables 44
6.5.2.1.4 Summary of Interlayer Interfaces . . . 44
6.5.2.2 Frame Transmission 45
6.5.2.3 Frame Reception 49
6.5.2.4 Network Management Interface Procedures . . 52
6.5.2.4.1 State Variable Initialization . . . 52
6.5.2.5 Common Procedures 54

7. ETHERNET PHYSICAL LAYER SPECIFICATION 55

7.1 Physical Channel Overview and Model 55
7.1.1 Channel Goals and Non-Goals 55
7.1.1.1 Goals 55
7.1.1.2 Non-Goals 55
7.1.2 Characteristics of the Channel 56
7.1.3 Functions Provided by the Channel 56
7.1.4 Implementation of the Channel 57
7.1.4.1 General Overview of Channel Hardware . . . 57
7.1.4.2 Compatibility Interfaces 57
7.1.5 Channel Configuration Model 58

7.1.6 Channel Interfaces 63

7.2 Transceiver Cable Compatibility Interface Specifications . . . 63
7.2.1 Transceiver Cable Signals 63
7.2.1.1 Transmit Signal 64
7.2.1.2 Receive Signal 64
7.2.1.3 Collision Presence Signal 64
7.2.1.4 Power 64
7.2.2 Transceiver Cable Parameters 65
7.2.2.1 Mechanical Configuration 65
7.2.2.2 Characteristic Impedance 65
7.2.2.3 Attenuation 65
7.2.2.4 Velocity of Propagation 65
7.2.2.5 Timing Distortion 65
7.2.2.6 Resistance 65
7.2.2.7 Transfer Impedance 65
7.2.3 Transceiver Cable Connectors 65
7.2.4 Transceiver Cable Drive 67
7.2.5 Transceiver Cable Receive 68
7.2.5.1 Load Impedance and Termination 68
7.2.5.2 Common Mode and CMRR 68
7.2.5.3 Transceiver Cable Squelch 68

7.3 Coaxial Cable Compatibility Interface Specifications 69
7.3.1 Coaxial Cable Component Specifications 69
7.3.1.1 Coaxial Cable Parameters 69
7.3.1.1.1 Characteristic Impedance 69
7.3.1.1.2 Attenuation 69
7.3.1.1.3 Velocity of Propagation 69
7.3.1.1.4 Mechanical Requirements 69
7.3.1.1.5 Timing Distortion 70
7.3.1.1.6 Jacket Marking 70
7.3.1.1.7 Transfer Impedance 70
7.3.1.1.8 DC Resistance 70
7.3.1.2 Coaxial Cable Connectors 70
7.3.1.3 Coaxial Cable Terminators 71
7.3.1.4 Transceiver-to-Coaxial Cable Connections . . 72
7.3.2 Coaxial Cable Signaling 72

7.4 Transceiver Specifications 73
7.4.1 Transceiver-to-Coaxial Cable Interface 73
7.4.1.1 Input Impedance 73
7.4.1.2 Bias Current 73
7.4.1.3 Transmit Output Levels 74
7.4.2 Transceiver-to-Transceiver Cable Interface 74
7.4.2.1 Transmit Pair 74
7.4.2.2 Receive Pair 74

7.4.2.3 Collision Presence Pair. 75
7.4.2.4 Power Pair 75
7.4.2.5 Transceiver-to-Transceiver Cable Connections . 75
7.4.3 Electrical Isolation 75
7.4.4 Reliability 76
7.4.5 Common Mode Current Shunt 76
7.4.6 Transmit Watchdog Timer 76
7.4.7 Collision Presence Test 76

7.5 Channel Logic 77
7.5.1 Channel Encoding 77
7.5.1.1 Encoder 77
7.5.1.2 Decoder 78
7.5.1.3 Preamble Generation 78
7.5.2 Collision Detect Signal 79
7.5.3 Carrier Sense Signal 79
7.5.4 Channel Framing 81
7.5.4.1 Beginning-of-Frame Sequence. 81
7.5.4.2 End-of-Frame Sequence. 81

7.6 Channel Configuration Requirements 81
7.6.1 Cable Sectioning 81
7.6.2 Transceiver Placement 82
7.6.3 System Earth Connection. 82
7.6.4 Repeaters 83
7.6.4.1 Carrier Detect and Transmit Repeat 83
7.6.4.2 Collision Detect and Collision Repeat. 83
7.6.4.3 Repeater Signal Regeneration 84
7.6.4.3.1 Signal Amplification 84
7.6.4.3.2 Signal Timing 84

7.7 Environment Specifications 84
7.7.1 Electromagnetic Environment 84
7.7.2 Temperature and Humidity 84

8. ETHERNET CONFIGURATION TESTING PROTOCOL 85

8.1 Goals 85
8.1.1 Configuration Testing Functions 85
8.1.2 Conformance Requirements. 85

8.2 Functions, Loopback Frames and Protocol Messages 86

8.3 Encapsulation of Loopback Protocol Frames for Transmission on an Ethernet 87

8.4 Frame and Message Formats 87
8.4.1 Reply Message 88
8.4.2 Forward Data Message 88

8.4.2.1 Restrictions on Forward Data Messages . . . 88

8.5 Operation of the Protocol 88

8.6 Usage Examples 90
8.6.1 Local Control Test Example 90
8.6.2 Remote Control Test Example 91

Appendices

APPENDIX A: GLOSSARY 92

APPENDIX B: ASSIGNMENT OF ADDRESS AND TYPE VALUES 95

APPENDIX C: CRC IMPLEMENTATION. 97

APPENDIX D: IMPLEMENTATION OF TRANSCEIVER CABLE DRIVER AND RECEIVER. 99

APPENDIX E: INTERFRAME RECOVERY 102

APPENDIX F: RECOGNITION OF MULTICAST GROUP ADDRESSES 103

Figures and Tables

Figure 4-1: Ethernet Architecture and Typical Implementation 8

Figure 4-2: Architectural Layering 9

Figure 4-3: Data Link Layer Functions 11

Figure 4-4: Physical Layer Functions 13

Figure 4-5: Architecture Including Network Management Interface. . . 14

Figure 6-1: Data Link Layer Frame Format 27

Figure 6-2: Structure of the Data Link Procedural Model 38

Figure 6-3: Control Flow Summary -- Client Layer Processes 39

Figure 6-4: Control Flow Summary -- Data Link Layer Processes . . . 40

Figure 7-1: Physical Channel Configurations 60

Table 7-1: Physical Channel Propagation Delay Budget 62

Figure 7-2: Maximum Transceiver Cable Transfer Impedance 66

Figure 7-3: Typical Transceiver Cable Waveform 67

Figure 7-4: Maximum Coaxial Cable Transfer Impedance 71

Figure 7-5: Typical Coaxial Cable Waveform 73

Figure 7-6: Manchester Encoding 78

Figure 7-7: Preamble Encoding 79
Figure 7-8: Functional Logic of **collisionDetect** Signal 80
Figure 7-9: Functional Logic of **carrierSense** Signal 80
Figure 8-1: A Loopback Frame Encapsulated in an Ethernet Packet . . 87
Figure 8-2: A Reply Message Encapsulated in a Forward Data Message . 89
Figure C-1: CRC Implementation 98
Figure D-1: Typical Transceiver Cable Driver 99
Figure D-2: Typical Transceiver Cable Receiver 100
Figure D-3: Shows Relationship Between Figure D-1 and Figure D-2 . . 100
Figure D-4: Typical Transceiver Block Diagram 101

1. INTRODUCTION

The Ethernet local area network provides a communication facility for high-speed data exchange among computers and other digital devices located within a moderate-sized geographic area. Its primary characteristics include:

Physical Layer:

Data rate: 10 Million bits/sec

Maximum station separation: 2.8 Kilometers

Maximum number of stations: 1024

Medium: Shielded coaxial cable, base band signalling

Topology: Branching non-rooted tree

Data Link Layer:

Link control procedure: Fully distributed peer protocol, with statistical contention resolution (CSMA/CD)

Message protocol: Variable size frames, "best-effort" delivery

The Ethernet, like other local area networks, falls in a middle ground between long-distance, low-speed networks that carry data for hundreds or thousands of kilometers, and specialized, very high speed interconnections that are generally limited to tens of meters. The Ethernet is intended primarily for use in such areas as office automation, distributed data processing, terminal access, and other situations requiring economical connection to a local communication medium carrying bursty traffic at high-peak data rates. Situations demanding resistance to hostile environments, real-time response guarantees, and so on, while not specifically excluded, do not constitute the primary environment for which the Ethernet is designed.

The precursor to the Ethernet specified in this document was the "Experimental Ethernet," designed and implemented by Xerox in 1975, and used continually since that time by thousands of stations. The Ethernet defined here builds on that experience, and on the larger base of the combined experience of Digital, Intel, and Xerox in many forms of networking and computer interconnection.

In specifying the Ethernet, this document provides precise detailed definitions of the lowest two layers of an overall network architecture. It thus defines what is generally referred to as a *link-level* facility. It does not specify the higher level protocols needed to provide a complete network architecture. Such higher level protocols would generally include such functions as internetwork communication, error recovery, flow control, security measures (e.g., encryption), and other higher

level functions that increase the power of the communication facility and/or tailor it to specific applications. In particular, it should be noted that all error recovery functions have been relegated to higher level protocols, in keeping with the low error rates that characterize local area networks.

One of the main objectives of this specification is *compatibility*. As stated in Section 3, it is intended that *every implementation of the Ethernet be able to exchange data with every other implementation*. It should be noted that higher level protocols raise their own issues of compatibility over and above those addressed by the Ethernet and other link-level facilities. This does not eliminate the importance of link-level compatibility, however. While the compatibility provided by the Ethernet does not guarantee solutions to higher level compatibility problems, it does provide a context within which such problems can be addressed, by avoiding low-level incompatibilities that would make direct communication impossible.

2. REFERENCES

The following three papers describe the Experimental Ethernet, and are reprinted in: "The Ethernet Local Network: Three Reports," Xerox Palo Alto Research Center Technical Report CSL-80-2 (February 1980).

[1] Metcalfe, R. M. and Boggs, D. R., "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM 19* 7 (July 1976).

[2] Crane, R. C. and Taft, E. A. "Practical Considerations in Ethernet Local Network Design," Presented at *Hawaii International Conference on System Sciences* (January 1980).

[3] Shoch, J. F. and Hupp, J. A. "Measured Performance of an Ethernet Local Network," Presented at *Local Area Communications Network Symposium* Boston (May 1979).

The following references describe the ISO Open Systems Model:

[4] Zimmermann, H., "OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communication COM-28* 4 (April 1980).

[5] International Organization for Standardization (ISO), "Reference Model of Open Systems Interconnection," *Document no. ISO/TC97/SC16 N227* (June 1979).

The following references describe the Pascal language (used in the Data Link Layer procedural model) and its derivative Concurrent Pascal:

[6] Jensen, K. and Wirth, N., *Pascal User Manual and Report, 2nd Edition.* Springer-Verlag (1974).

[7] Brinch Hansen, P., *Concurrent Pascal Report.* Technical Report CIT-IS-TR 17, California Institute of Technology (1975).

The following references discuss the CRC code used for the frame check sequence:

[8] Hammond, J. L., Brown, J. E. and Liu, S. S., "Development of a Transmission Error Model and an Error Control Model," Technical Report RADC-TR-75-138, Rome Air Development Center (1975).

[9] Bittel, R., "On Frame Check Sequence (FCS) Generation and Checking," ANSI working paper X3-S34-77-43, (1977).

The following references are cited as support information for the electrical standards to which the specifications given in Section 7 comply.

[10] International Electrotechnical Commission, IEC Recommendation, Publication 435, Safety of Data Processing Equipment, First Edition, 1973.

[11] Underwriters Laboratories, UL 478 Standard for Electronic Data-Processing Units and Systems, Fourth Edition, April 1980.

[12] National Fire Protection Association, National Electrical Code.

3. GOALS AND NON-GOALS

This section states the assumptions underlying the design of the Ethernet.

3.1 Goals

The goals of the Ethernet design are:

Simplicity: Features which would complicate the design without substantially contributing to the meeting of the other goals have been excluded.

Low cost: Since technological improvements will continue to reduce the overall cost of stations wishing to connect to the Ethernet, the cost of the connection itself should be minimized.

Compatibility: All implementations of the Ethernet should be capable of exchanging data at the data link level. For this reason, the specification avoids optional features, to eliminate the possibility of incompatible variants of the Ethernet.

Addressing flexibility: The addressing mechanisms should provide the capability to target frames to a single station, a group of stations, or to all stations on the network.

Fairness: All stations should have equal access to the network when averaged over time.

Progress: No single station operating in accordance with the protocol should be able to prevent the progress of other stations.

High speed: The network should operate efficiently at a data rate of 10 Megabits per second.

Low delay: At any given level of offered traffic, the network should introduce as little delay as possible in the transfer of a frame.

Stability: The network should be stable under all load conditions, in the sense that the delivered traffic should be a monotonically non-decreasing function of the total offered traffic.

Maintainability: The Ethernet design should allow for network maintenance, operation, and planning.

Layered Architecture: The Ethernet design should be specified in layered terms to separate the logical aspects of the data link protocol from the physical details of the communication medium.

3.2 Non-Goals

The following are *not* goals of the Ethernet design:

Full duplex operation: At any given instant, the Ethernet can transfer data from one source station to one or more destination stations. Bi-directional communication is provided by rapid exchange of frames, rather than full duplex operation.

Error control: Error handling at the data link level is limited to detection of bit errors in the physical channel, and the detection and recovery from collisions. Provision of a complete error control facility to handle detected errors is relegated to higher layers of the network architecture.

Security: The data link protocol does not employ encryption or other mechanisms to provide security. Higher layers of the network architecture may provide such facilities as appropriate.

Speed flexibility: This specification defines a physical channel operating at a single fixed data rate of 10 Megabits per second.

Priority: The data link protocol provides no support of priority station operation.

Hostile user: There is no attempt to protect the network from a malicious user at the data link level.

4. FUNCTIONAL MODEL OF THE ETHERNET ARCHITECTURE

There are two important ways to view the Ethernet design, corresponding to:

Architecture, emphasizing the logical divisions of the system, and how they fit together.

Implementation, emphasizing the actual components, and their packaging and interconnection.

Figure 4-1 illustrates these two views as they apply to a typical implementation, showing how each view groups the various functions.

This document is organized along *architectural* lines, emphasizing the large-scale separation of the Ethernet system into two parts: the *Data Link Layer* and the *Physical Layer*. These layers are intended to correspond closely to the lowest layers of the ISO Model for Open Systems Interconnection [4,5]. Architectural organization of the specification has two main advantages:

Clarity: A clean overall division of the design along architectural lines makes the specification clearer.

Flexibility: Segregation of medium-dependent aspects in the Physical Layer allows the Data Link Layer to apply to transmission media other than the specified coaxial cable.

As is evident in Figure 4-1, the architectural model is based on a set of interfaces different from those emphasized in the implementations. One crucial aspect of the design, however, must be addressed largely in terms of the implementation interfaces: *compatibility*. Two important compatibility interfaces are defined within what is architecturally the Physical Layer:

Coaxial cable interface: To communicate via the Ethernet, all stations must adhere rigidly to the exact specification of coaxial cable signals defined in this document, and to the procedures which define correct behavior of a station. The medium-independent aspects of the Data Link Layer should not be taken as detracting from this point: *communication via the Ethernet requires complete compatibility at the coaxial cable interface.*

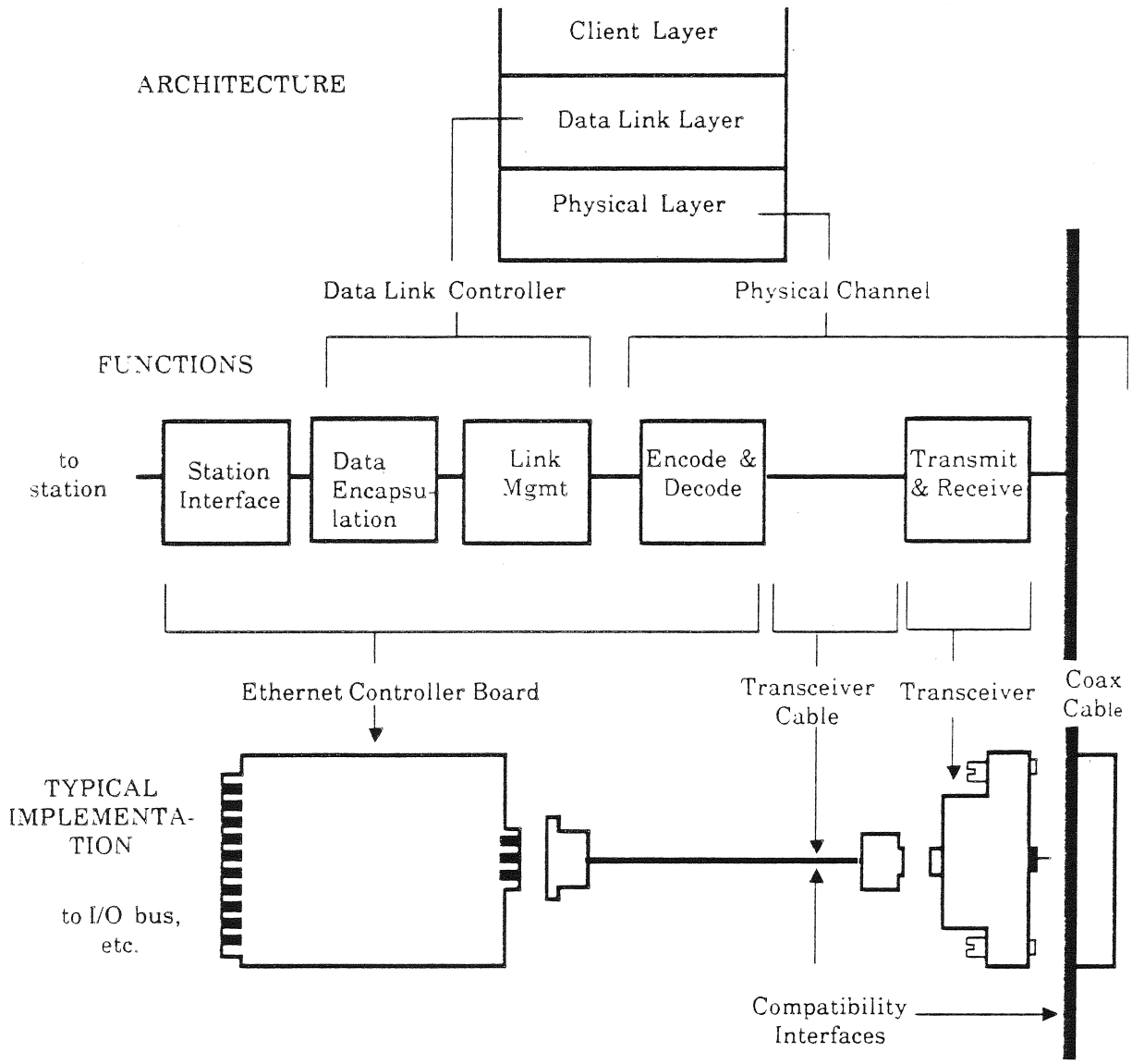


Figure 4-1: Ethernet Architecture and Typical Implementation

Transceiver cable interface: It is anticipated that most stations will be located some distance away from their connection to the coaxial cable. While it is necessary to place a small amount of circuitry (the *transceiver*) directly adjacent to the coaxial cable, the majority of the electronics (the *controller*) can and should be placed with the station. Since it is desirable for the same transceiver to be usable with a wide variety of stations, a second compatibility interface, the *transceiver cable interface*, is defined. While conformance with this interface is not strictly necessary to ensure communication, it is highly recommended, since it allows maximum flexibility in intermixing transceivers and stations.

4.1 Layering

The major division in the Ethernet Architecture is between the Physical Layer and the Data Link Layer, corresponding to the lowest two levels in the ISO model. The higher levels of the overall network architecture, which use the Data Link Layer, will be collectively referred to in this document as the "Client Layer" since, strictly speaking, the identity and function of higher level facilities are outside the scope of this specification. The one exception is a distinguished client of the Data Link Layer called the Network Management System, for which this document specifies a special interface and a Configuration Testing Protocol. The intent is that the Ethernet Physical and Data Link Layers support the higher layers of the ISO model (Network Layer, Transport Layer, etc.).

The basic structure of the layered architecture is shown in Figure 4-2.

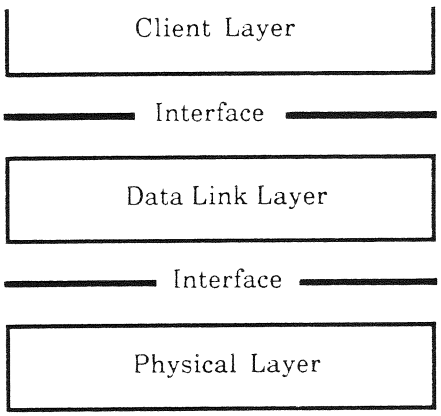


Figure 4-2: Architectural Layering

In the architectural model used here, the layers interact via well-defined interfaces.

The interface between the Client Layer and the Data Link Layer includes facilities for transmitting and receiving frames, and provides per-operation status information for use by higher-level error recovery procedures.

The interface between the Data Link Layer 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.

The interface for network management includes facilities required for interfacing a Network Management System. These facilities allow the observation and control of the status, configuration, and operation of the Data Link and Physical Layers.

These interfaces are described more precisely in Section 5.

4.2 Data Link Layer

The Data Link Layer defines a medium-independent link level communication facility, built on the medium-dependent physical channel provided by the Physical Layer. It is applicable to a general class of local area broadcast media suitable for use with the channel access discipline known as carrier-sense multiple-access with collision-detection (CSMA-CD). Compatibility with non-contention media (e.g., switched lines, token-passing rings, etc.), while a worthwhile topic for further research, is not addressed in this specification.

The Data Link Layer specified here is intended to be as similar as possible to that described in the ISO model. In a broadcast network like the Ethernet, the notion of a data link between two network entities does not correspond directly to a distinct physical connection. Nevertheless, the two main functions generally associated with a data link control procedure are present:

Data encapsulation/decapsulation

- framing (frame boundary delimitation)

- addressing (handling of source and destination addresses)

- error detection (detection of physical channel transmission errors)

Link management

- channel allocation (collision avoidance)

- contention resolution (collision handling)

This split is reflected in the division of the Data Link Layer into the Data Encapsulation sub-layer and the Link Management sub-layer (see Figure 4-3).

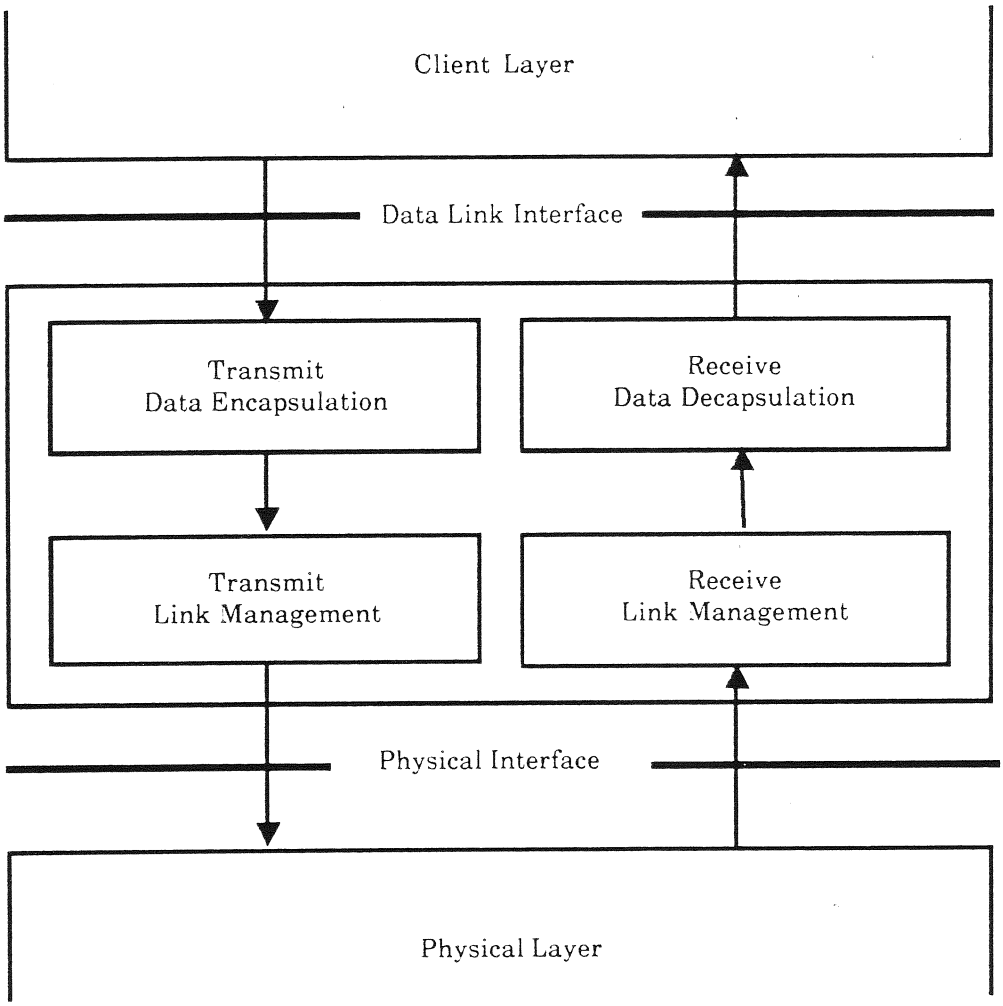


Figure 4-3: Data Link Layer Functions

In terms of the ISO model, the Ethernet Data Link Layer provides a multi-endpoint connection between higher-layer entities wishing to communicate. The connection provided is called a *data link*, and is implemented between two or more Data Link Layer entities called *data link controllers* via a Physical Layer connection called the *physical channel.*

4.3 Physical Layer

The Physical Layer specified in this document provides a 10 Mbps physical channel through a coaxial cable medium. Because one purpose of the layered architecture is to insulate the Data Link Layer from the medium-specific aspects of the channel, the Physical Layer completely specifies the essential physical characteristics of the Ethernet, such as data encoding, timing, voltage levels, etc. Implementation details are left unspecified, to retain maximum flexibility for the implementer. In all cases, the criterion applied in distinguishing between essential characteristics and implementation details is guaranteed compatibility: any two correct implementations of the Physical Layer specified here will be capable of exchanging data over the coaxial cable, enabling communication between their respective stations at the Data Link Layer.

The Physical Layer defined in this specification performs two main functions generally associated with physical channel control:

Data encoding

- preamble generation/removal (for synchronization)
- bit encoding/decoding (between binary and phase-encoded form)

Channel access

- bit transmission/reception (of encoded data)
- carrier sense (indicating traffic on the channel)
- collision detection (indicating contention on the channel)

This split is reflected in the division of the Physical Layer into the Data Encoding sub-layer and the Channel Access sub-layer, as shown in Figure 4-4.

4.4 Network Management

The Ethernet Data Link Layer provides an interface to a *Network Management System.* A Network Management System provides services which are necessary for network planning, operations, and maintenance.

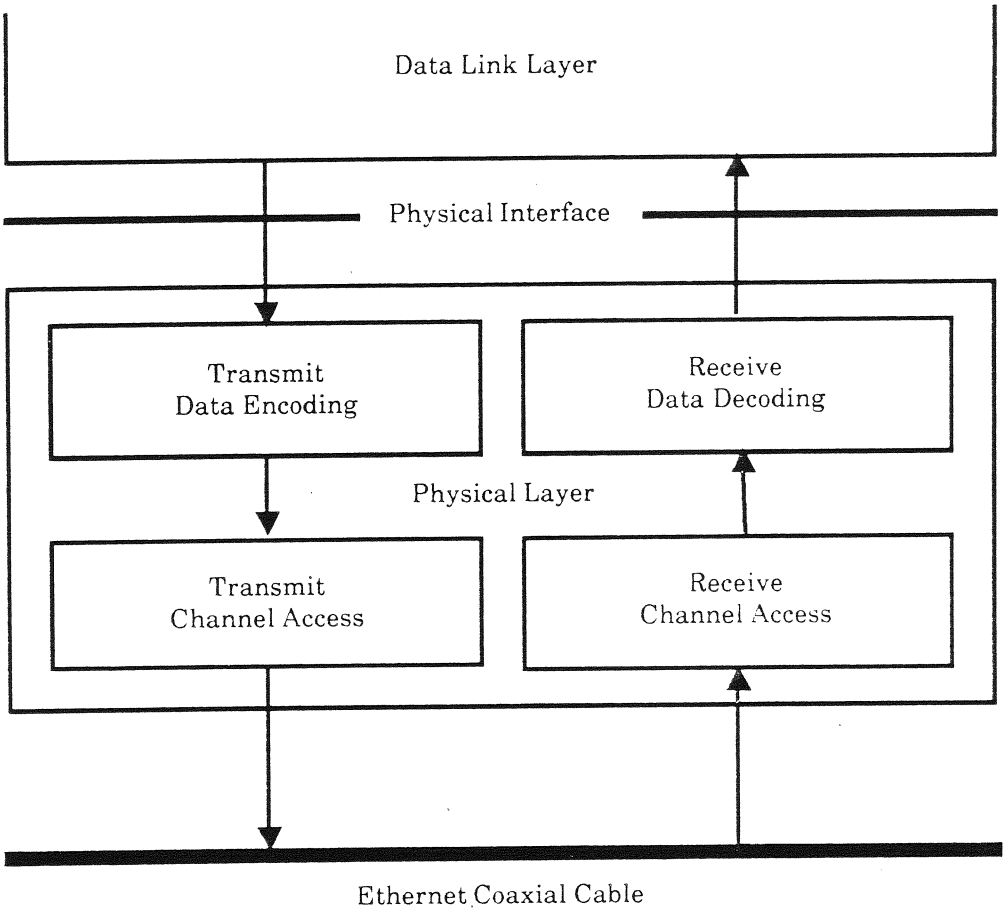


Figure 4-4: Physical Layer Functions

Network *planning* gathers usage information to help the users determine when and how to expand the network.

Network *operations* deal with normal, day-to-day functions, such as initialization, monitoring, and performance optimization.

Network *maintenance* deals with the detection, isolation, and repair of faults.

Figure 4-5 shows how a Network Management System is positioned with respect to the other layers of the Ethernet in an overall network architecture. It uses two interfaces to the Data Link Layer: 1) the general interface used by all other clients, and 2) a dedicated one called the Network Management Interface. In addition, a Network Management System usually will have access to each one of the higher layers of the network architecture, in many cases via a dedicated interface. Details of such access are beyond the scope of this specification.

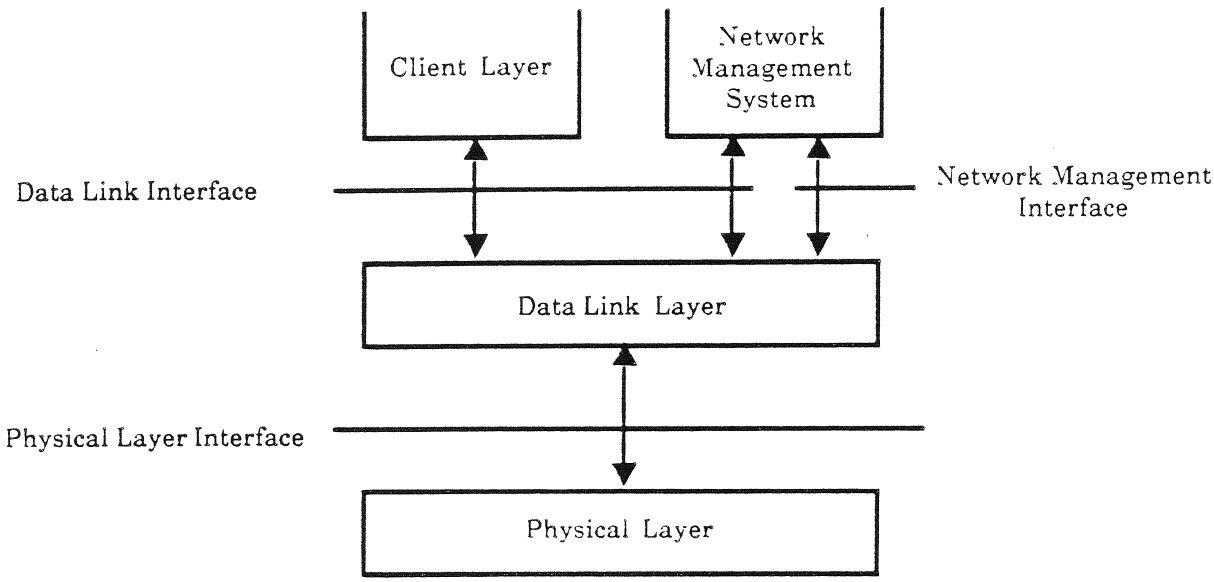


Figure 4-5: Architecture Including Network Management Interface

The interface between the Network Management System and the Ethernet Data Link Layer is specified in Sections 5 and 6. Through this interface, Network Management can perform two main functions:

Configuration control

- initiating, suspending, and resuming Data Link operation
- setting the station's physical address
- setting the station's addressing modes (normal, promiscuous, multicast)

Observation of

- the values of certain data link state variables and parameters
- activity data (frames sent and received, bad frames, collisions)
- error conditions (operation of carrier sense and collision detect)

Using the general Client-to-Data Link Interface and the Configuration Testing protocol defined in Section 8, Network Management can ascertain the operability of other stations on the network.

4.5 Ethernet Operation and the Functional Model

This section provides an overview of frame transmission, frame reception, and the interface to Network Management, in terms of the functional model of the architecture. This overview is descriptive, rather than definitional; the formal specifications of the operations described here are given in Sections 6 and 7.

4.5.1 Transmission Without Contention

When the Client Layer requests the transmission of a frame, the Transmit Data Encapsulation component of the Data Link Layer constructs the frame from the client-supplied data and appends a frame check sequence to provide for error detection. The frame is then handed to the Transmit Link Management component for transmission.

Transmit Link Management attempts to avoid contention with other traffic on the channel by monitoring the *carrier sense* signal and *deferring* to passing traffic. When the channel is clear, frame transmission is initiated (after a brief interframe delay to provide recovery time for other data link controllers and for the physical channel). The Data Link Layer then provides a serial stream of bits to the Physical Layer for transmission.

The Data Encoding component of the Physical Layer, before sending the actual bits of the frame, sends an encoded preamble to allow the receivers and repeaters along the channel to synchronize their clocks and other circuitry. It then begins translating the bits of the frame into encoded form and passes them to the Channel Access component for actual transmission over the medium.

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

When transmission has completed without contention, the Data Link Layer so informs the Client Layer and awaits the next request for frame transmission.

4.5.2 Reception Without Contention

At the receiving station, the arrival of a frame is first detected by the Receive Channel Access component of the Physical Layer, 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 passed to the Receive Data Decoding component.

Receive Data Decoding translates the encoded signal back into binary data and discards the leading bits, up to and including the end of the preamble. It then passes subsequent bits up to the Data Link Layer.

Meanwhile, the Receive Link Management component of the Data Link Layer, having seen carrier sense go on, has been waiting for the incoming bits to be delivered. Receive Link Management collects bits from the Physical Layer as long as the carrier sense signal remains on. When the carrier sense signal goes off, the frame is 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 contents of the frame to the Client Layer along with an appropriate status code. The status code is generated 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.

4.5.3 Collisions: Handling of Contention

If multiple stations attempt to transmit at the same time, it is possible for their transmitting data link controllers to interfere with each other's transmissions, in spite of their attempts to avoid this by deferring. When two stations' transmissions overlap, the resulting contention is called a *collision*. A given station can experience a collision during the initial part of its transmission (the "collision window"), before its transmitted signal has had time to propagate to all parts of the Ethernet channel. Once the collision window has passed, the station is said to have *acquired* the channel; subsequent collisions are avoided, since all other (properly functioning) stations can be assumed to have noticed the signal (via carrier sense) and to be deferring to it. The time to acquire the channel is thus based on the *round-trip propagation time* of the physical channel.

In the event of a collision, the Transmit Channel Access component of a transmitting station's Physical Layer first notices the interference on the channel and turns on the collision detect signal. This is noticed in turn by the Transmit Link Management component of the Data Link Layer, and collision handling begins. First, Transmit Link Management *enforces* the collision by transmitting a bit sequence called the *jam*. This insures that the duration of the collision is sufficient to be noticed by the other transmitting station(s) involved in the collision. After the jam is sent, Transmit Link Management terminates the transmission and schedules a retransmission attempt for a randomly selected time in the near future. Retransmission is attempted repeatedly in the face of repeated collisions. Since repeated collisions indicate a busy channel, however, Transmit Link Management attempts to adjust to the channel load by *backing off* (voluntarily delaying its own retransmissions to reduce its load on the channel). This is accomplished by expanding the interval from which the random retransmission time is selected on each retransmission attempt. Eventually, either the transmission succeeds, or the attempt is abandoned on the assumption

that the channel has failed or has become overloaded.

At the receiving end, the bits resulting from a collision are received and decoded by the Physical Layer just as are the bits of a valid frame. The Physical Layer is not required to assert the collision detect signal during frame reception; however, the assertion of the collision detect signal indicates a true collision in the Physical Layer. Instead, the fragmentary frames received during collisions are distinguished from valid frames by the Data Link's Receive Link Management component, by noting that a collision fragment is always smaller than the shortest valid frame. Such fragments are discarded by Receive Link Management.

4.5.4 Functional Description of Network Management Interface

The features presented here provide a minimum set of capabilities for the Ethernet Network Management Interface. They do not preclude an implementation which embellishes these services or add new ones. However, it guarantees that all Ethernet implementations possess a minimum set of primitives which will allow the control and observation of the operation of an installed network.

The interface provides these services for network management:

- Initiate, Suspend and Resume operations

  Network Management can initiate, suspend, and resume the operation of the Data Link Interface. Suspension never interrupts a frame transmission or reception that is already in progress; it takes effect immediately after the operation in progress is completed. The operation of the Network Management Interface cannot be suspended.

- Read or Set addresses

  Read and set address services provide network managers with configuration and control features. Network Management can read and set the station's physical address. The Data Link uses this address as the source address on transmission and as the physical destination address on reception. This address must be assigned in conformance with the procedures given in Appendix B.

- Read or Set addressing modes

  Network Management can change certain modes of operation of Ethernet stations. There are two addressing modes: promiscuous mode which accepts all frames received regardless of destination address, and normal mode which accepts only frames with the broadcast address, the station's physical address, or, when enabled, multicast addresses.

- Enable and disable multicast operation

  In general, multicast addresses are assigned by a local network administrator, following procedures described in Appendix B. Network Management can enable and disable the reception of frames sent to multicast addresses. However, reception of frames sent to the broadcast address (see Section 6.2.1) cannot be deactivated.

- Observe Data Link activity

  The Data Link maintains counters of data link activity, such as frames sent and received. These counters can be read and re-initialized by Network Management.

- Check operation of physical channel

  The Ethernet Data Link maintains two flags indicating, when raised, that a malfunction of the carrier sense or collision detect mechanisms has been observed. Only Network Management can reset these flags. A Network Management System monitoring these flags can log the state of the Data Link in the event of these failures.

5. INTER-LAYER INTERFACES

The purpose of this section is to provide precise definitions of the interfaces between the architectural layers defined in Section 4. In order to provide such a definition, some precise notation must be adopted. The notation used here is the Pascal language, in keeping with the procedural nature of the formal Data Link Layer specification (see 6.5). Each interface is thus described as a set of procedures and/or shared variables which 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 6.5, which 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 Client Layer and the Data Link Layer, and need not be reflected in the implementation interface between a station and its controller.

5.1 Client Layer to Data Link Layer

The two primary services provided to the Client Layer by the Data Link Layer are transmission and reception of frames. The interface through which the Client Layer uses the facilities of the Data Link Layer therefore consists of a pair of functions.

Functions:

TransmitFrame
ReceiveFrame

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

The Client Layer transmits a frame by invoking *TransmitFrame*:

```
function TransmitFrame (
    destinationParam: AddressValue;
    sourceParam: AddressValue;
    typeParam: TypeValue;
    dataParam: DataValue): TransmitStatus;
```

The TransmitFrame operation is synchronous, in the sense that its duration is the entire attempt to transmit the frame, so that when the operation completes, transmission has either succeeded or failed, as indicated by the resulting status

code:

```
type TransmitStatus = (transmitOkNoCollision, transmitOkOneCollision,
transmitOkMultipleCollisions, excessiveCollisionError, lateCollisionError,
dataLinkOff);
```

Successful transmission is indicated by any of the following status codes: *transmitOkNoCollision*, *transmitOkOneCollision*, *transmitOkMultipleCollisions*. The code *excessiveCollisionError* indicates that the transmission attempt was aborted due to excessive collisions, because of heavy traffic or a network failure; the code *lateCollisionError* indicates that a collision was detected outside the collision window (implementation of the lateCollisionError status code and associated processing is optional); the code *dataLinkOff* indicates that transmission was not attempted because the Data Link was turned off by network management (see section 5.3), or never turned on. Implementations may define additional implementation-dependent status codes if necessary.

The Client Layer accepts incoming frames by invoking *ReceiveFrame*:

```
function ReceiveFrame (
  var destinationParam: AddressValue;
  var sourceParam: AddressValue;
  var typeParam: TypeValue;
  var dataParam: DataValue): ReceiveStatus;
```

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

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

Successful reception is indicated by the status code *receiveOK*. The code *frameCheckError* indicates that the frame received was damaged by a transmission error in the physical channel. The code *alignmentError* indicates that the frame received was damaged, and that in addition, its length was not an integral number of octets. The code *dataLinkOff* indicates that reception was not possible because the Data Link was turned off by network management, or never turned on. Implementations may define additional implementation-dependent status codes if necessary.

5.2 Data Link Layer to Physical Layer

The interface through which the Data Link Layer uses the facilities of the Physical Layer consists of a function, a pair of procedures and three Boolean

variables.

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

During transmission, the contents of an outgoing frame are passed from the Data Link Layer to the Physical Layer via repeated use of the *TransmitBit* operation:

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

Each invocation of *TransmitBit* passes one new bit of the outgoing frame to the Physical Layer. The *TransmitBit* operation is synchronous, in the sense that the duration of the operation is the entire transmission of the bit, so that when the operation completes, the Physical Layer is ready to accept the next bit immediately. (Note: this does not imply that all invocations of *TransmitBit* are of exactly equal duration; for example, if the Physical Layer must perform some initial processing–e.g., preamble generation–before transmitting the first bit of a frame, the first invocation of *TransmitBit* may take significantly longer.)

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

```
var transmitting: Boolean;
```

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

The presence of a collision in the physical channel is signaled to the Data Link Layer via the variable *collisionDetect*:

```
var collisionDetect: Boolean;
```

The *collisionDetect* signal remains *true* for the duration of the collision. (Note: Since an entire collision may occur during the first invocation of *TransmitBit*–e.g., during preamble generation–the Data Link Layer must handle this possibility by monitoring *collisionDetect* concurrently with its transmission of outgoing bits. See 6.5 for details.)

The *collisionDetect* signal is also used to implement the self-test described in 7.4.7. The Data Link Layer must continue to monitor *collisionDetect* for 2 μs after the transmission is finished. The collisionDetect signal is in an undefined state during the 4 μs inhibit period, as defined in 7.5.3, following frame reception. The

precise operation of the collisionDetect signal is specified in 7.5.2.

During reception, the contents of an incoming frame are retrieved from the Physical Layer by the Data Link Layer via repeated use of the *ReceiveBit* operation:

```
function ReceiveBit: Bit;
```

Each invocation of *ReceiveBit* retrieves one new bit of the incoming frame (i.e., not including any preamble bits) from the Physical Layer. The *ReceiveBit* operation is synchronous, in the sense that its duration is the entire reception of a single bit. (As with *TransmitBit*, the first invocation of *ReceiveBit* may take significantly longer--e.g., due to preamble removal). Upon receiving a bit, the Data Link Layer must immediately request the next bit until all bits of the frame have been received. (See 6.5 for details.)

The overall event of data being transmitted on the physical channel is signaled to the Data Link Layer via the variable *carrierSense*:

```
var carrierSense: Boolean;
```

When the Physical Layer sets *carrierSense* to *true* the Data Link Layer must immediately begin retrieving the incoming bits via the *ReceiveBit* operation. When *carrierSense* subsequently becomes *false*, the Data Link Layer 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 Data Link Layer. (See 6.5 for details.)

The Data Link Layer must monitor the value of *carrierSense* to defer its own transmissions when the channel is busy, as well as during its own transmissions in order to ascertain the proper operation of the carrier sense function.

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 Data Link Layer to measure time intervals in units of the (physical-channel-dependent) bit time.

Another important property of the Physical Layer, which is an implicit part of the interface presented to the Data Link Layer, is the maximum *round-trip propagation time* of the physical channel. This figure 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 specified to be at most 464 bit times (see 7.1.2).

5.3 Data Link to Network Management Interface

This interface permits the Network Management System to observe and control the operation of the Data Link Layer. This interface consists of a set of variables and one procedure. Some of these variables affect the operation of the Data Link Layer, while others inform Network Management of various conditions, including some failures of the Physical Layer. The procedure initializes the Data Link.

Variables:
- *dataLinkOn*
- *physicalAddress*
- *multicastOn*
- *addressMode*
- *framesSentNoErrors*
- *framesReceivedNoErrors*
- *framesAbortedExcessCollisions*
- *framesReceivedCRCErrors*
- *framesReceivedAlignErrors*
- *framesAbortedLateCollision* (optional)
- *carrierSenseFailed*
- *collisionDetectFailed*

Procedure *Initialize*

All of these parameters are described as variables in the procedural model, but they can be grouped into categories according to their use. The first four of them are Data Link state variables that can be read and set by Network Management. Then there are five counters: the first two have 32 bits and the other three have 16 bits. In addition, there is an optional 16 bit counter. The last two variables in the list are flags.

The initial values of the Data Link parameters are set by the Data Link procedure Initialize when the Data Link begins operation (see 6.5.2.4.1).

The Boolean dataLinkOn is used by Network Management to suspend and resume the transmission and reception of frames by the Data Link Layer:

```
var dataLinkOn: Boolean;
```

Setting dataLinkOn to false is different from turning off the station; it does not interrupt a reception or transmission that is already in progress. Instead, the

inhibition of the TransmitFrame and ReceiveFrame operations takes place after completion of the operation in progress. All other Data Link Layer processes remain active, and Network Management can still read and set all the other variables available in the interface. Analogously, setting dataLinkOn to true merely resumes the suspended operation of TransmitFrame and ReceiveFrame: this is different from initializing the Data Link, which is done by the procedure Initialize.

Each Ethernet station has a unique 48-bit address assigned to it, in conformance with the address assignment procedures described in Appendix B. The Network Management System is able to read and set this address. The parameter *physicalAddress* denotes this address.

```
var physicalAddress: AddressValue;
```

When frames are received from the Physical Layer, the function *RecognizeAddress* compares the value of *destinationAddress* in the received frame (see Section 5.2) with the value *physicalAddress*. The operation of this procedure also depends on the values of the following two variables:

```
var multicastOn: Boolean;

var addressMode: ReceptionMode;
```

The first variable enables or disables multicast address recognition; the second variable can cause all incoming frames to be accepted:

```
type ReceptionMode = (normal, promiscuous);
```

There are five (optionally, six) counters. Incremented by the Data Link Layer, these counters can be individually read and reset by Network Management. These counters are implemented such that upon attaining their maximum permissible value, they remain at that value, regardless of further increment operations, until reset. The counter names are explanatory of their use:

```
type

  Counter32 = 0..4294967295;

  Counter16 = 0..65535;

var

  framesSentNoErrors, framesReceivedNoErrors: Counter32;

  framesAbortedExcessCollisions, framesAbortedLateCollision,
  framesReceivedCRCErrors, framesReceivedAlignErrors: Counter16;
```

```
framesAbortedLateCollision: Counter16; {optional}
```

The Data Link Layer informs Network Management of failures in the operation of the Physical Layer by setting the following flags:

```
var carrierSenseFailed, collisionDetectFailed: Boolean;
```

Two Data Link Layer processes, CarrierSenseTest and CollisionDetectTest, are responsible for setting these flags to true if a malfunction occurs. Only Network Management can reset them to false. A Network Management System may include processes that watch these flags in order to capture the state of the Data Link when they are set.

The Data Link Layer also provides Network Management with the procedure Initialize:

```
procedure Initialize;
```

This procedure defines initial values for certain state variables in the Data Link and resets Network Management counters.

6. ETHERNET DATA LINK LAYER SPECIFICATION

6.1 Data Link Layer Overview and Model

As defined in Section 4, the Ethernet Architecture consists of the Data Link Layer, and below it, the Physical Layer. Furthermore, the Data Link Layer is divided into two sub-layers (see Figure 4-3).

Data encapsulation
- framing
- addressing
- error detection

Link management
- channel allocation
- contention resolution

This model is used throughout this section to structure the detailed specification of the Data Link Layer. An English description of the Data Link Layer is given in 6.2, 6.3, and 6.4. A more precise algorithmic definition is given in 6.5, which provides a procedural model for the Data Link Layer in the form of a program in the language Pascal. Note that whenever there is any apparent ambiguity concerning the definition of some aspect of the Data Link Layer, it is the Pascal procedural specification in 6.5 which should be consulted for the definitive statement.

6.2 Frame Format

The data encapsulation function of the Data Link Layer comprises the construction and processing of frames. The subfunctions of framing, addressing, and error detection are reflected in the frame format as follows:

Framing: No explicit framing information is needed, since the necessary framing cues (*carrierSense* and *transmitting*) are present in the interface to the Physical Layer.

Addressing: Two address fields are provided to identify the source and destination stations for the frame.

Error detection: A Frame Check Sequence field is provided for detection of transmission errors.

Figure 6-1 shows the five fields of a frame: the addresses of the frame's source and destination, a type field for use by higher layers (see 6.2.2), a data field containing the transmitted data, and the frame check sequence field containing a cyclic redundancy check value to detect transmission errors. Of these five fields, all are of fixed size except the data field, which may contain any integral number of octets

between the minimum and maximum values specified below (see 6.2.5).

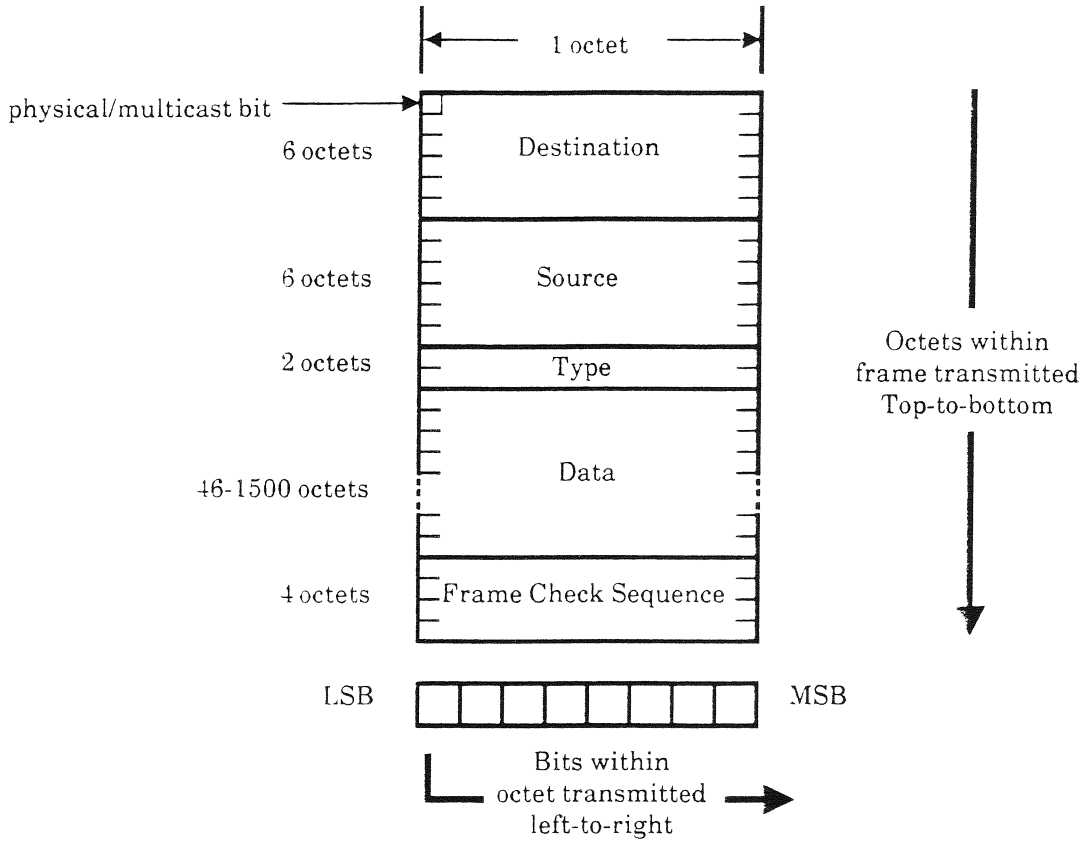


Figure 6-1: Data Link Layer Frame Format

Relative to Figure 6-1, the octets of a frame are transmitted from top to bottom, and the bits of each octet are transmitted from left to right.

NOTE

With the exception of Section 8, this document does *not* define an order of transmission for the octets of standard multi-octet data types (strings, integers, etc.), since no values of such data types appear in the data link frame format. The order in which implementations of the Ethernet store the octets of a frame in computer memory, and the manner in which higher level protocols interpret the contents of the data field as values of various multi-octet data types, are beyond the scope of this specification.

The Ethernet itself is also totally insensitive to the interpretation of bits within an octet as constituting the digits of an 8-digit binary numeric value. Since some uniform convention is helpful, however, in avoiding needless incompatibility among different station types, the interpretation is arbitrarily defined to be that

the left-most bit (first transmitted) is the low-order (2^0) digit and the right-most bit (last transmitted) is the high-order (2^7) digit.

6.2.1 Address Fields

Data link addresses are 6 octets (48 bits) in length. There are two types of data link addresses.

Physical address: The unique address associated with a particular station on the Ethernet. A station's physical address should be distinct from the physical address of any other station on *any* Ethernet. Physical addresses of Ethernet stations must be displayed in human readable form, by either a tag or label, an LED display, a properly documented software command, or some other well-defined method.

Multicast address: A multi-destination address, associated with one or more stations on a given Ethernet. There are two kinds of multicast addresses:

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

- *Broadcast address:* A distinguished, predefined multicast address which always denotes the set of *all* stations on a given Ethernet. The broadcast address should be used only when strictly necessary: excessive broadcasting of packets can overload some stations.

The first bit of a data link address distinguishes physical from multicast addresses:

0 ⇒ physical address

1 ⇒ multicast address

In either case, the remainder of the first octet and all of the subsequent octets form a 47-bit pattern. In the case of the broadcast address, this pattern consists of 47 one-bits. There is no standard "null "address value.

When in human-readable form, Ethernet addresses must be written and displayed in the following format: 12 hexadecimal digits, with the digits paired in groups of 2. These digit pairs (representing octets) shall be separated by single hyphens. The order of transmission on the Ethernet shall be from the leftmost octet (as written or displayed) to the rightmost octet. The order of bits within the octets shall be from the least significant bit of the rightmost digit to the most significant bit of the leftmost digit. For example, the following Ethernet Address:

F0-2E-15-6C-77-9B

corresponds to the following sequence of bits on the Ethernet:

0000 1111 0111 0100 1010 1000 0011 0110 1110 1110 1101 1001

where the bits are transmitted from left to right.

The procedures for assigning unique values for physical and multicast addresses are discussed in Appendix B.

6.2.1.1 Destination Address Field

The destination address field specifies the station(s) for which the frame is intended. It may be a physical or multicast (including broadcast) address. For details of address recognition by the receiving station(s), see 6.4.1.2.

6.2.1.2 Source Address Field

The source address field contains the physical address of the station sending the frame. This field is not interpreted at the Data Link Layer. It is specified at the data link level because a uniform convention for the placement of this field is crucial for most higher level protocols.

Client layers use the Data Link physical address for the source address of all frames transmitted.

6.2.2 Type Field

The type field consists of a two-octet value reserved for use by higher levels (in particular, to identify the Client Layer protocol associated with the frame). The type field is uninterpreted at the Data Link Layer. It is specified at this level because a uniform convention for the placement and value assignment of this field is crucial if multiple higher level protocols are to be able to share the same Ethernet network without conflict. Appendix B discusses the assignment of type field values.

6.2.3 Data Field

The data field contains a sequence of n octets, where $46 < n < 1500$ Within this range, full data transparency is provided, in the sense that any arbitrary sequence of octet values may appear in the data field.

6.2.4 Frame Check Sequence Field

The frame check sequence (FCS) field contains a 4-octet (32-bit) cyclic redundancy check (CRC) value. This value is computed as a function of the contents of the source, destination, type and data fields (i.e., all fields except the frame check sequence field itself). The encoding is defined by the generating polynomial:

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

(This polynomial has been used in the Autodin-II network; its properties are investigated in [8].)

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.

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

Appendix C discusses CRC implementation issues.

6.2.5 Frame Size Limitations

Given the limitations on the size of the data field specified in 6.2.3 and the 18 octet total size for the other four fields, the smallest valid frame contains 64 octets and the largest valid frame contains 1518 octets.

6.3 Frame Transmission

Frame transmission includes data encapsulation and link management aspects:

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

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

The performance of these functions by a transmitting data link controller interacts with corresponding actions by other data link controllers to jointly implement the

Ethernet data link protocol.

6.3.1 Transmit Data Encapsulation

6.3.1.1 Frame Assembly

The fields of the data link frame are set to the values provided by the Client Layer as arguments to the *TransmitFrame* operation (see 5.1), with the exception of the frame check sequence, which is set to the CRC value generated by the Data Link.

6.3.1.2 Frame Check Sequence Generation

The CRC value defined in 6.2.4 is generated and inserted in the frame check sequence field, following the fields supplied by the Client Layer. Appendix C discusses CRC implementation.

6.3.2 Transmit Link Management

6.3.2.1 Carrier Deference

Even when it has nothing to transmit, the Data Link monitors the physical channel for traffic by watching the *carrierSense* signal provided by the Physical Layer. Whenever the channel is busy, the Data Link *defers* to the passing frame by delaying any pending transmission of its own. After the last bit of the passing frame (i.e., when *carrierSense* changes from true to false), the Data Link continues to defer for 9.6 μs minimum to provide proper interframe spacing (see 6.3.2.2). At the end of that time, if it has a frame waiting to be transmitted, transmission is initiated independent of the value of *carrierSense*. When transmission has completed, the Data Link resumes its original monitoring of *carrierSense*. (If there was nothing to transmit, the Data Link resumes its original monitoring of carrierSense immediately.)

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

6.3.2.2 Interframe Spacing

As defined in 6.3.2.1, the rules for deferring to passing frames insure a minimum interframe spacing of 9.6 μs. This is intended to provide interframe recovery time for other data link controllers and for the physical channel.

Note that 9.6 μs is the minimum value of the interframe spacing. If necessary for implementation reasons, a transmitting controller may use a larger value with a resulting decrease in its maximum throughput. The value should not exceed 10.6 μs.

6.3.2.3 Collision Handling

Once the Data Link has finished deferring and has started transmission, it is still possible for it to experience contention for the channel. As discussed in 4.4.3, collisions can occur until acquisition of the network has been accomplished through the deference of all other data link controllers.

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

- It is an upper bound on the acquisition time of the network.
- It is an upper bound on the length of a frame fragment generated by a collision. (See 6.4.2.1)
- It is the scheduling quantum for retransmission. (See 6.3.2.3.2)

In order to fulfill all three functions, the slot time must be larger than the sum of the Physical Layer round-trip propagation time (less than 464 bit times; see 7.1.2) and the Data Link Layer maximum jam time (48 bit times, see 6.3.2.3.1). The slot time is defined to be 512 bit times.

6.3.2.3.1 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 at least 32 (but not more than 48) additional bits have been transmitted (counting from the time *collisionDetect* went on). This collision enforcement or "jam" guarantees that the duration of the collision is sufficient to ensure its detection by all transmitting stations on the network. The content of the jam is unspecified; it may be any fixed or variable pattern convenient to the data link controller implementation, but should not be designed to be the 32-bit CRC value corresponding to the (partial) frame transmitted prior to the jam.

6.3.2.3.2 Collision Backoff and Retransmission

When a transmission attempt has terminated due to a collision, it is retried by the transmitting Data Link until either it is successful, or 16 attempts (the original attempt plus 15 retries) 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 Data Link delays before attempting to retransmit the frame. The delay is an integral multiple of the slot time. (See 6.3.2.3.) The number of slot

times to delay before the n^{th} retransmission attempt, is equal to a uniformly distributed random integer r in the range $0 \leq r < 2^k$ where $k = \min(n, 10)$. If all 16 attempts fail, this event is reported as an error.

Note that the values given above define the most aggressive behavior that a station may exhibit in attempting to retransmit after a collision. In the course of implementing the retransmission scheduling procedure, a station may introduce extra delays which 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.

6.4 Frame Reception

Frame reception includes both data decapsulation and link management aspects:

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

Receive Link Management's main function is the filtering of collision fragments from complete incoming frames.

The performance of these functions by a receiving data link controller interacts with corresponding actions by other data link controllers to jointly implement the Ethernet data link protocol.

6.4.1 Receive Data Decapsulation

6.4.1.1 Framing

The Data Link recognizes the boundaries of an incoming frame by monitoring the *carrierSense* signal provided by the Physical Layer. There are two possible length errors that can occur, which indicate ill-framed data: the frame may be too long, or its length may not be an integral number of octets.

6.4.1.1.1 Maximum Frame Size

The receiving Data Link is not required to enforce the frame size limit specified in 6.2.5, but it is allowed to truncate frames longer than 1518 octets and report this event as an (implementation-dependent) error.

6.4.1.1.2 Integral Number of Octets in Frame

Since the format of a valid frame specifies an integral number of octets, only a collision or an error can produce a frame with a length that is not an integral multiple of 8. Complete frames (i.e., not rejected as collision fragments; see 6.4.2.1) that do not contain an integral number of octets are truncated to the nearest octet boundary. If frame check sequence validation (see 6.4.1.3) detects an

error in such a frame, the status code *alignmentError* is reported.

6.4.1.2 Address Recognition

The Ethernet Data Link is capable of recognizing physical and multicast addresses, as defined in 6.2.1.

6.4.1.2.1 Physical Addresses

The Data Link recognizes and accepts any frame whose destination field contains the physical address of the station.

The physical address of each station is set by network management to a unique value associated with the station, and distinct from the address of any other station on any Ethernet. The setting of the station's physical address by network management allows multiple data link controllers connected to a single station to respond to the same physical address. The procedures for allocating unique addresses are discussed in Appendix B.

6.4.1.2.2 Multicast Addresses

The Data Link recognizes and accepts any frame whose destination field contains the broadcast address.

The Data Link is capable of activating multicast-group address recognition when requested by Network Management. When so activated, the Data Link recognizes and accepts any frame whose destination field contains a multicast-group address. Multicast-group address recognition may be similarly deactivated by Network Management. Processing of individual multicast addresses is the responsiblity of the client. (See Appendix F.)

6.4.1.3 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 is reported as such. Implementation issues are discussed in Appendix C.

6.4.1.4 Frame Disassembly

The frame is disassembled and the fields are passed to the Client Layer via the output parameters of the *ReceiveFrame* operation (see 5.1).

6.4.2 Receive Link Management

6.4.2.1 Collision Filtering

As specified in 6.2.5, the smallest valid frame must contain at least 64 octets. Any

frame containing less than 64 octets is presumed to be a fragment resulting from a collision and is discarded by the receiving Data Link. Since occasional collisions are a normal part of the link management procedure, the discarding of such a fragment is not reported as an error to the Client Layer.

6.5 The Data Link Layer Procedural Model

6.5.1 Overview of the Procedural Model

The functions of the Ethernet Data Link Layer are presented below, modeled as a program written in the language Pascal [6]. This procedural model is intended as the primary specification of the functions to be provided in any Ethernet Data Link Layer implementation. It is important to distinguish, however, between the model and a real implementation. The model is optimized for simplicity and clarity of presentation, while any realistic implementation must place heavier emphasis on such constraints as efficiency and suitability to a particular implementation technology or computer architecture. In this context, several important properties of the procedural model must be considered.

6.5.1.1 Ground Rules for the Procedural Model

a) First, it must be emphasized that the description of the Data Link Layer in a programming language is in no way intended to imply that a data link controller must be implemented as a program executed by a computer. The implementation may consist of any appropriate technology including hardware, firmware, software, or any combination.

b) Similarly, it must be emphasized that it is the *behavior* of Data Link Layer implementations that must match the specification, *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.

c) 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 Data Link Layer and are only serialized for presentation to the Physical Layer. In reality, many data link controller implementations will instead handle frames serially on a bit, octet or word basis. A serial implementation would typically perform the required functions (address recognition, frame check sequence generation/validation, etc.) in an overlapped, pipelined fashion. This approach has not been reflected in the procedural model, since this would only complicate the description of the functions without changing them in any way.

d) The model consists of algorithms designed to be executed by a number of concurrent processes; these algorithms collectively implement the Ethernet data link control procedure. The timing dependencies introduced by the need

for concurrent activity are resolved in two ways:

Processes vs. 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 data link procedure *ReceiveFrame* is always called well before the frame in question has started to arrive.

Processes vs. Processes: Among processes, no assumptions are made about relative speeds of execution. This means that each interaction between two processes must be structured to work correctly independent of their respective speeds. Note, however, that the timing of interactions among processes is often, in part, an indirect reflection of the timing of external events, in which case appropriate timing assumptions may still be made.

It is intended that the concurrency in the model reflect the parallelism intrinsic to the task of implementing the Ethernet data link, although the actual parallel structure of the implementations is likely to vary.

6.5.1.2 Use of Pascal in the Procedural Model

Pascal was chosen for the procedural model because of its relative simplicity and clarity, and its general acceptance.

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

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

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

   2) The process and cycle constructs of the Pascal derivative Concurrent Pascal [7] 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.

   3) 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). In fact, 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.

4) The use of a variant record to represent a frame (both as fields and as bits) follows the letter but not the spirit of the Pascal Report, since it allows the underlying representation to be viewed as two different data types. (It also assumes that this representation is as shown in Figure 6-1.)

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

6.5.2 Procedural Model

The procedural model used here is based on seven cooperating concurrent processes. Of these, five are actually defined in the Data Link Layer. The remaining two processes are provided by the Client Layer and utilize the interface operations provided by the Data Link Layer. The seven processes are thus:

Client Layer:

Frame Transmitter Process **Frame Receiver Process**

Data Link Layer:

Bit Transmitter Process **Bit Receiver Process**
Deference Process
Collision Presence Test Process **Carrier Sense Test Process**

This organization of the model is illustrated in Figure 6-2, and reflects the fact that the communication of entire frames is initiated by the Client Layer, while the timing of collision backoff and of individual bit transfers is based on interactions between the Data Link Layer and the Physical-Layer-dependent bit-time.

Figure 6-2 depicts the static structure of the procedural model, showing how the various processes and procedures interact by invoking each other. Figures 6-3 and 6-4 summarize the dynamic behavior of the model during transmission and reception, focusing on the steps that must be performed, rather than the procedural structure which performs them. The usage of the shared state variables is not depicted in the figures, but is described in the comments in 6.5.2.1. Figure 6-3 does not show the detailed operation of the transmitStatus codes or the handling of the optional lateCollisionError.

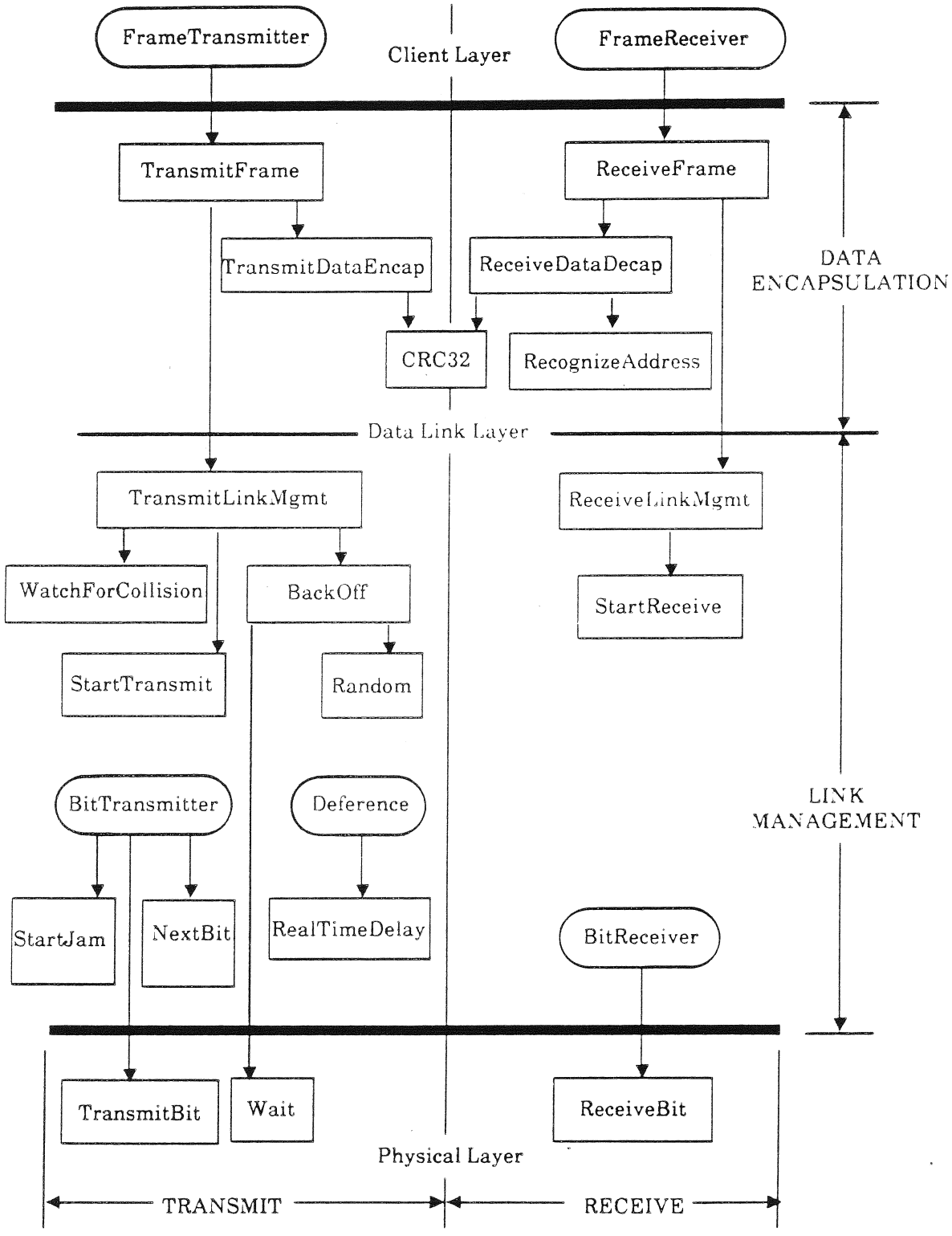


Figure 6-2: Structure of the Data Link Procedural Model

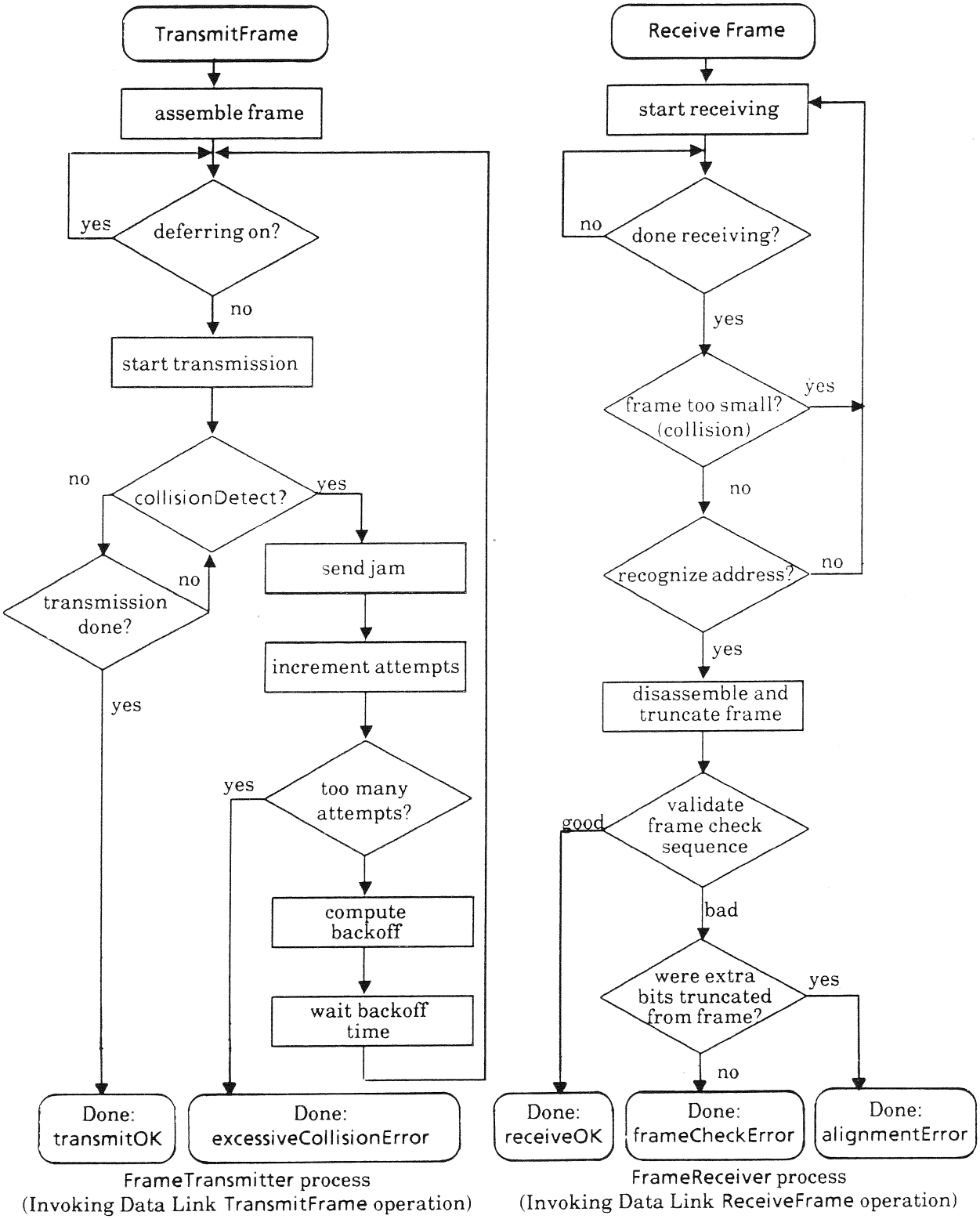


Figure 6-3: Control Flow Summary -- Client Layer Processes

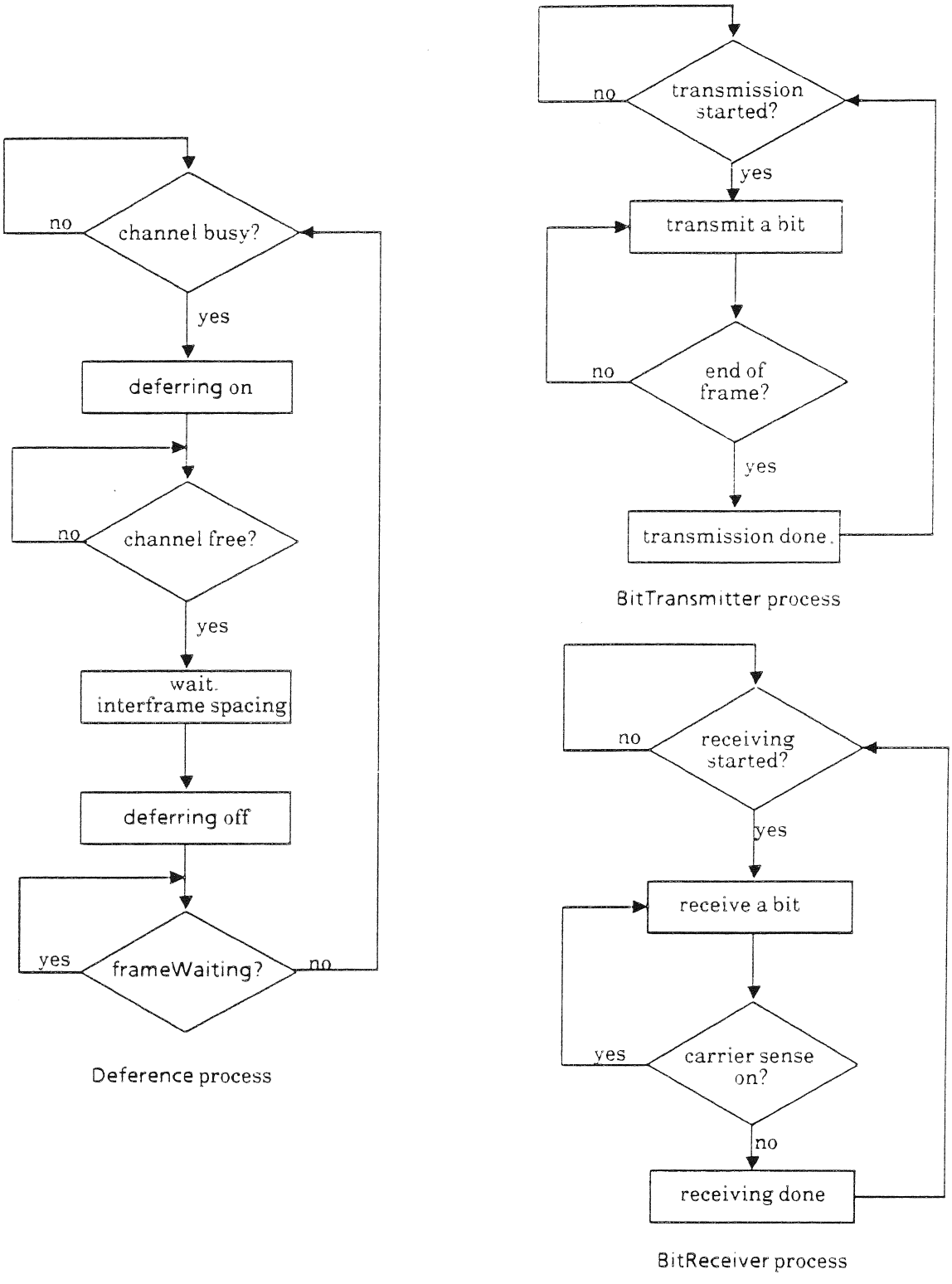


Figure 6-4: Control Flow Summary -- Data Link Layer Processes

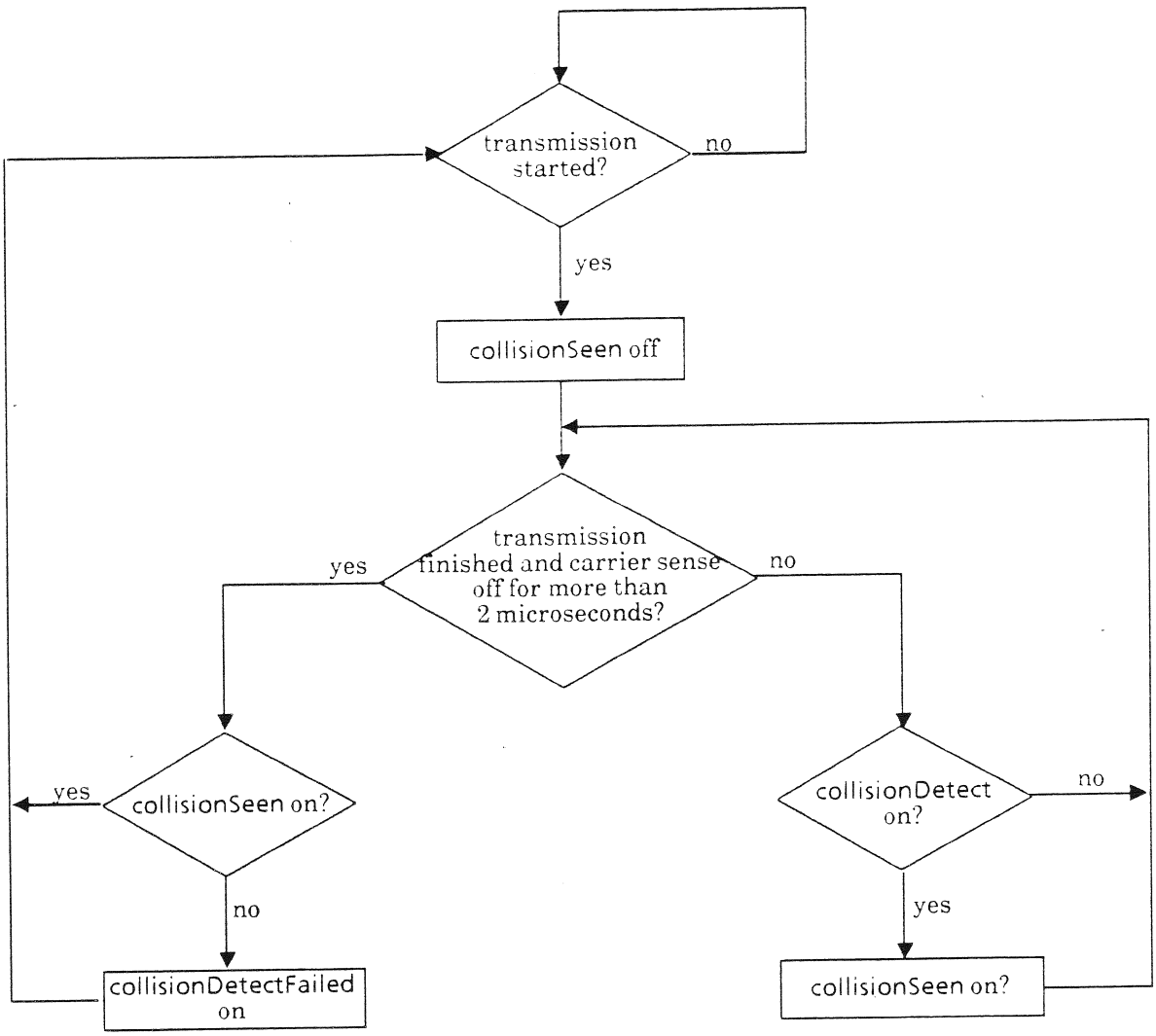
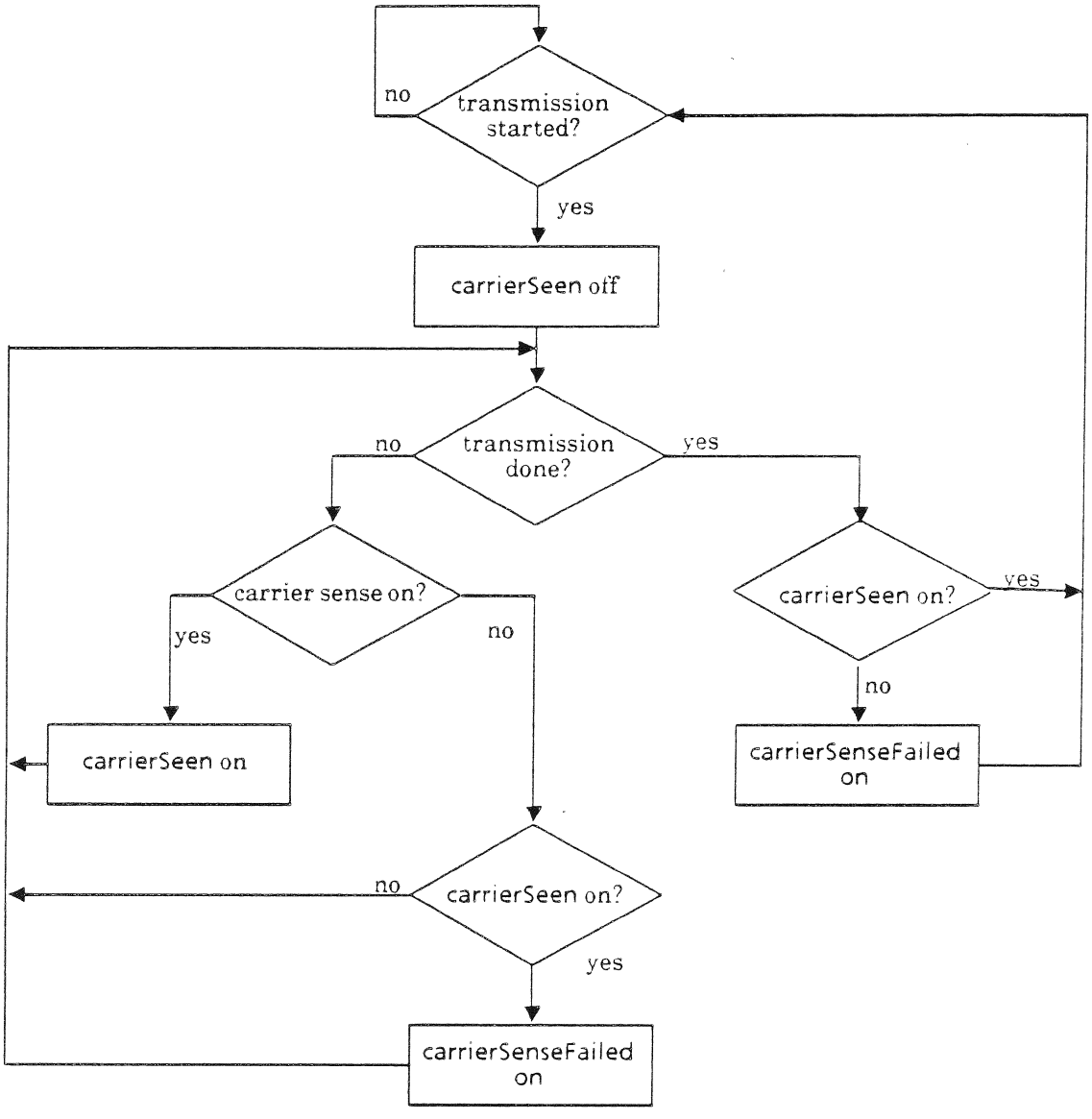


Figure 6-4: Control Flow Summary -- Data Link Layer Processes (Contd)



CarrierSense Test Process

Figure 6-4: Control Flow Summary -- Data Link Layer Processes (Contd)

6.5.2.1 Global Declarations

6.5.2.1.1 Common Constants and Types

The following declarations of constants, types and variables are used by the frame transmission and reception sections of each data link controller:

```
const
    addressSize = 48;  {48 bit address = 6 octets}
    typeSize = 16;  {16 bit protocol type = 2 octets}
    dataSize = ...;  {see 6.5.1.2, note 3}
    crcSize = 32;  {32 bit CRC = 4 octets}
    frameSize = ...;  { = 2*addressSize + typeSize + dataSize + crcSize...see 6.5.1.2, note 3}
    broadcastAddress = ...; {constant value consisting of 48 1's, recognized by all stations}
    slotTime = 512;  {unit of time for collision handling}

type
    Bit = 0..1;
    AddressValue = array [1..addressSize] of Bit;
    TypeValue = array [1..typeSize] of Bit;
    DataValue = array [1..dataSize] of Bit;
    CRCValue = array [1..crcSize] of Bit;

    ViewPoint = (fields, bits);  {Two ways to view the contents of a frame}

    Frame = record  {Format of data link frame}
        case view: ViewPoint of
            fields: (
                destinationField: AddressValue;
                sourceField: AddressValue;
                typeField: TypeValue;
                dataField: DataValue;
                fcsField: CRCValue);
            bits: (
                contents: array [1..frameSize] of Bit)
        end; {Frame}
```

6.5.2.1.2 Transmit State Variables

The following items are specific to frame transmission. (See also 6.5.2.1.4 on interfaces.)

```
const
    interFrameSpacing = 9.6;  {minimum time between frames, in microseconds}
    attemptLimit = 16;  {Max number of times attempt transmission}
    backOffLimit = 10;  {Limit on number of times to back off}
    jamSize = 32;  {jam may be 32 to 48 bits long}

var
```

```
outgoingFrame: Frame; {The frame to be transmitted
currentTransmitBit, lastTransmitBit: 1..frameSize; {Positions of current and last outgoing
 bits in outgoingFrame}
deferring: Boolean; {True implies any pending transmission must wait for the channel to clear}
frameWaiting: Boolean; {Indicates that outgoingFrame is deferring}
attempts: 0..attemptLimit; {Number of transmission attempts on outgoingFrame}
lateCollision: Boolean; {optional, indicates that collision has occurred outside the collision window}
newCollision: Boolean; {Indicates that a collision has occurred but has not yet been jammed}
transmitSucceeding: Boolean; {Running indicator of whether transmission is succeeding}
```

6.5.2.1.3 Receive State Variables

The following items are specific to frame reception. (See also 6.5.2.1.4 on interfaces)

```
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}
```

6.5.2.1.4 Summary of Interlayer Interfaces

The interface to the Client Layer, defined in 5.1, is summarized below:

```
type
    TransmitStatus = (transmitOkNoCollision, transmitOkOneCollision,
     transmitOkMultipleCollisions, excessiveCollisionError, lateCollisionError,
     dataLinkOff); {Result of TransmitFrame operation}

    ReceiveStatus = (receiveOK, frameCheckError, alignmentError,
     dataLinkOff); {Result of ReceiveFrame operation}

function TransmitFrame (
    destinationParam: AddressValue;
    sourceParam: AddressValue;
    typeParam: TypeValue;
    dataParam: DataValue): TransmitStatus; {Transmits one frame}

function ReceiveFrame (
    var destinationParam: AddressValue;
    var sourceParam: AddressValue;
    var typeParam: TypeValue;
    var dataParam: DataValue): ReceiveStatus; {Receives one frame}
```

The interface to the Physical Layer, defined in 5.2, is summarized below:

```
var
    carrierSense: Boolean; {Indicates incoming bits}
    transmitting: Boolean; {Indicates outgoing bits}
    collisionDetect: Boolean; {Indicates channel 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}
```

The interface to Network Management, defined in 5.3, is summarized below:

```
const
    maxCounter32Value = 4294967295; {2^32 - 1}
    maxCounter16Value = 65535; {2^16 - 1}

type
    ReceptionMode = (normal, promiscuous); {changes mode of the Data Link function
     RecognizeAddress}
    Counter32 = 0..maxCounter32Value;
    Counter16 = 0..maxCounter16Value;

var
    dataLinkOn: Boolean; {enable/disable frame transmission and reception}
    physicalAddress: AddressValue; {the current station address for Data Link Layer operation}
    multicastOn: Boolean; {controls multicast operation}
    addressMode: ReceptionMode; {controls the mode used for address recognition}
    framesSentNoErrors, framesReceivedNoErrors: Counter32; {32-bit counters}
    framesAbortedExcessCollisions, framesReceivedCRCErrors,
      framesReceivedAlignErrors: Counter16; {16-bit counters}
    framesAbortedLateCollision: Counter16; {optional 16-bit counter}
    carrierSenseFailed, collisionDetectFailed: Boolean; {flags to monitor activity of carrier
      sense and collision detect logic}

procedure Initialize; {used by Network Management to initialize Data Link operation}
```

6.5.2.2 Frame Transmission

The algorithms in this section define data link frame transmission.

The function *TransmitFrame* implements the frame transmission operation provided to the Client Layer:

```
function TransmitFrame (
    destinationParam: AddressValue;
    sourceParam: AddressValue;
    typeParam: TypeValue;
```

```
    dataParam: DataValue): TransmitStatus;
  procedure TransmitDataEncap; ... {nested procedure; see body below}
begin
  if dataLinkOn then
  begin
    TransmitDataEncap;
    TransmitFrame : = TransmitLinkMgmt;
  end
  else TransmitFrame : = dataLinkOff
end; {TransmitFrame}
```

First, *TransmitFrame* calls the internal procedure *TransmitDataEncap* to construct the frame. It then calls *TransmitLinkMgmt* 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;
    typeField : = typeParam;
    dataField : = dataParam;
    fcsField : = CRC32(outgoingFrame);
    view : = bits
  end {assemble frame}
end; {TransmitDataEncap}
```

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;
  lateCollision : = false;
  while attempts < attemptLimit and not transmitSucceeding
    and not lateCollision do
  begin {loop}
    if attempts > 0 then BackOff;
    frameWaiting : = true;
```

```
    while deferring do nothing;  {defer to passing frame, if any}
    frameWaiting : = false;
    StartTransmit;
    while transmitting do WatchForCollision;
    attempts : = attempts + 1
  end; {loop}
  if transmitSucceeding then
  begin
    IncrementCounter32(framesSentNoErrors);
    if attempts = 0 then TransmitLinkMgmt : = transmitOkNoCollision;
    if attempts = 1 then TransmitLinkMgmt : = transmitOkOneCollision;
    if attempts > 1 then TransmitLinkMgmt : = transmitOkMultipleCollisions
  end
  else if lateCollision then
  begin
    IncrementCounter16(framesAbortedLateCollision);
    TransmitLinkMgmt : = lateCollision
  end
  else
  begin
    IncrementCounter16(framesAbortedExcessCollisions);
    TransmitLinkMgmt: = excessiveCollisionError;
  end
end; {TransmitLinkMgmt}
```

The *IncrementCounter* procedures are defined in 6.5.2.4.

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
end; {StartTransmit}
```

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

```
procedure WatchForCollision;
begin
  if transmitSucceeding and collisionDetect then
  begin
    newCollision : = true;
    transmitSucceeding : = false;
```

```
    if currentTransmitBit > slotTime then
    lateCollision : = true; {This is optional}
  end
end; {WatchForCollision}
```

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

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

```
var maxBackOff: 2..1024; {Working variable of BackOff}

procedure BackOff;
begin
  if attempts = 1 then maxBackOff : = 2 else if attempts ≤ backOffLimit
    then maxBackOff : = maxBackOff*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}
    while carrierSense do nothing; {wait for carrier to disappear}
    RealTimeDelay(interFrameSpacing);
    deferring : = false; {allow new transmissions to proceed}
    while frameWaiting do nothing {allow waiting transmission (if any)}
  end {main loop}
end; {Deference}

procedure RealTimeDelay (usec: real);
  var timeOut: real;
begin
  timeOut : = RealTimeClock + usec;
```

```
    while RealTimeClock ≤ timeOut do nothing; {wait for the specified number of μsec}
  end; {RealTimeDelay}

  function RealTimeClock: real;
  begin
    {real time clock in μsec}
  end; {RealTimeClock}
```

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

```
  process BitTransmitter;
  begin
    cycle {outer loop}
      while transmitting do
      begin {inner loop}
        TransmitBit(outgoingFrame[currentTransmitBit]); {send next bit to Physical Layer}
          if newCollision then StartJam else NextBit
      end {inner loop}
    end {outer loop}
  end; {BitTransmitter}

  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 may contain 32 to 48 bits of arbitrary data. (*StartJam* uses the first 32 bits of the frame, merely to simplify this program).

6.5.2.3 Frame Reception

The algorithms in this section define data link frame reception:

The procedure *ReceiveFrame* implements the frame reception operation provided to the Client Layer:

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

```
    var sourceParam: AddressValue;
    var typeParam: TypeValue;
    var dataParam: DataValue): ReceiveStatus;
  function ReceiveDataDecap: ReceiveStatus; ... {nested function; see body below}
begin
  if dataLinkOn then;
  repeat
    ReceiveLinkMgmt;
    ReceiveFrame : = ReceiveDataDecap;
  until receiveSucceeding;
  else ReceiveFrame : = dataLinkOff
end; {ReceiveFrame}
```

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

```
function ReceiveDataDecap: ReceiveStatus;
begin
  receiveSucceeding : = RecognizeAddress
  (incomingFrame.destinationField);
if receiveSucceeding then with incomingFrame do
begin {disassemble frame}
  view : = fields;
  destinationParam : = destinationField;
  sourceParam : = sourceField;
  typeParam : = typeField;
  dataParam : = dataField;
  if fcsField = CRC32(incomingFrame) then
    begin
      IncrementCounter32(framesReceivedNoErrors);
      ReceiveDataDecap : = receiveOk;
    end
    else if excessBits = 0 then
    begin
       IncrementCounter16(framesReceivedCRCErrors);
       ReceiveDataDecap : = frameCheckError;
    end
    else
    begin
       IncrementCounter16(framesReceivedAlignErrors);
       ReceiveDataDecap : = alignmentError;
    end;
    view : = bits
```

```
  end {disassemble frame}
end; {ReceiveDataDecap}
```

The Boolean function *RecognizeAddress* is used by the Data Link Layer to decide which frames to receive:

```
function RecognizeAddress(address:AddressValue):Boolean;
begin
  if address = physicalAddress or addressMode = promiscuous
  or address = broadcastAddress
  then RecognizeAddress: = true
 else if multicastOn and address[1] = 1
    then RecognizeAddress : = true
    else RecognizeAddress : = false
end; {RecognizeAddress}
```

RecognizeAddress returns true in the following cases: 1) *addressMode* is *promiscuous;* 2) *address* matches the value of *physicalAddress*; 3) *address* matches the value of *broadcastAddress*; 4) *multicastOn* is true and address is a multicast address.

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 ≥ slotTime); {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 channel 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}
      b : = ReceiveBit;  {Get next bit from physical link}
      if carrierSense then
      begin {append bit to packet}
        incomingFrame[currentReceiveBit] : = b;
        currentReceiveBit : = currentReceiveBit + 1
      end; {append bit to packet}
      receiving : = carrierSense
    end {inner loop}
  end {outer loop}
end; {BitReceiver}
```

6.5.2.4 Network Management Interface Procedures

The algorithms in this section specify how the Data Link Layer maintains the variables that constitute the interface to Network Management.

6.5.2.4.1 State Variable Initialization

The procedure *Initialize* must be run when the Data Link Layer begins operation, before any of the processes begin execution:

```
procedure Initialize;
begin
  dataLinkOn : = false; {temporarily, until all values are reset}
    frameWaiting : = false;
    deferring : = false;
    newCollision : = false;
    transmitting : = false; {In interface to Physical Layer; see below}
    receiving : = false;
    framesSentNoErrors : = 0;
    framesReceivedNoErrors : = 0;
    framesAbortedExcessCollisions : = 0;
    framesReceivedCRCErrors : = 0;
    framesReceivedAlignErrors : = 0;
    framesAbortedLateCollision : = 0; {optional}
    carrierSenseFailed : = false;
    collisionDetectFailed : = false;
    physicalAddress : = ...; {implementation-specific value}
    addressMode : = ...; {implementation-specific value}
    multicastOn : = ...; {implementation-specific value}
    dataLinkOn : = true;
```

```
  while carrierSense do nothing
    {Start execution of all processes}
  end; {Initialize}
```

Initialize sets certain crucial shared state variables to their initial values (all other global variables are appropriately reinitialized before each use), waits for the channel to be idle, and starts operation of the various processes.

The two *IncrementCounter* procedures increase the counters when they have not reached their maximum possible values:

```
  procedure IncrementCounter32(var counter: Counter32);
  begin
    if counter < maxCounter32Value then counter : = counter + 1;
  end; {IncrementCounter32}

  procedure IncrementCounter16(var counter: Counter16);
  begin
    if counter < maxCounter16Value then counter : = counter + 1;
  end; {IncrementCounter16}
```

These two procedures are used in the Data Link Layer functions *TransmitLinkMgmt* and *ReceiveDataDecap*.

The Data Link Layer process *CarrierSenseTest* sets the *carrierSenseFailed* flag if *carrierSense* disappears while transmitting or if it never appears during an entire transmission.

```
  process CarrierSenseTest;
    var carrierSeen :Boolean; {running indicator of whether carrierSense has been true at any time during
    the current transmission}
  begin
    cycle {main loop}
      while not transmitting do nothing; {wait for start of transmission}
      carrierSeen : = false;
      while transmitting do
        if carrierSense then carrierSeen : = true
        else if carrierSeen then carrierSenseFailed : = true; {it disappeared}
      if not carrierSeen then carrierSenseFailed : = true {it never appeared}
      end {main loop}
  end; {CarrierSenseTest}
```

After transmitting goes true, there may be some delay until the carrierSense signal is set by the Physical Layer.

The collision presence test (see 7.4.7) is implemented by the Data Link Layer process CollisionDetectTest:

```
process CollisionDetectTest;
  var collisionSeen : Boolean; timeOut : real;
begin
  cycle {main loop}
    collisionSeen : = false;
    while not transmitting do nothing; {wait for start of transmission}
    while transmitting or carrierSense do
      if collisionDetect then collisionSeen : = true;
    timeOut : = RealTimeClock + 2.0; {i.e., 2 μsec}
    while RealTime Clock ≤ timeOut do
      if collisionDetect then collisionSeen : = true;
    if not collisionSeen then collisionDetectFailed : = true
  end {main loop}
end; {CollisionDetectTest}
```

This process monitors the *collisionDetect* signal during the entire transmission as well as for 2 μsec following the deassertion of carrierSense after the end of transmission. If the transmission resulted in a collision, the correct operation of the collision detect circuitry can be inferred because it informed the Data Link Layer of the collision. If the transmission ended without collisions and the collisionDetect signal does not come in time, the collisionDetectFailed flag is set to inform Network Management that the collision presence test specified in 7.4.7 failed.

6.5.2.5 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 as defined in 6.2.4}
end; {CRC32}
```

To enhance readability, the following procedure is also defined:

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

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

7. ETHERNET PHYSICAL LAYER SPECIFICATION: Baseband Coaxial System

7.1 Physical Channel Overview and Model

The Ethernet physical channel (henceforth referred to as the channel) provides the lowest layer in the Ethernet architecture. It performs all the functions needed to transmit and receive data at the physical level, while supporting the Data Link Layer to Physical Layer Interface described in 5.2.

This section describes the requirements for interface and compatibility with a baseband coaxial implementation of the channel.

7.1.1 Channel Goals and Non-goals

This section states the objectives underlying the design of the channel.

7.1.1.1 Goals

The following are the goals of the channel:

- Provide a means for communication between Ethernet data link entities.
- Define physical interfaces so that hardware manufacturers' implementations are compatible.
- Provide all clocks, synchronization, and timing required for both the channel and the Ethernet data link.
- Provide high bandwidth and low bit error rates.
- Provide for ease of installation and service.
- Provide for high network availability.
- Support the Ethernet Data Link Layer to Physical Layer Interface.
- Low cost.

7.1.1.2 Non-Goals

The following are not goals of the baseband coaxial channel design:

- Operation at data rates other than 10 Mbps per second.
- Operation with media other than the specified coaxial cable.
- Simultaneous use of the channel by transmitters using signals not specified in

this document.

- Protection against a malicious user or a malfunctioning data link entity is not provided by the channel as specified. However, higher layers (above the data link) and/or physical security means may be employed to achieve this.

7.1.2 Characteristics of the Channel

The channel provides (and the data link assumes) the following characteristics:

- The ability to send and receive information (non-simultaneously) between any two or more data link entities on the same network.
- The ability to detect the presence of another station's transmission while not transmitting (*carrier sense*).
- The ability to detect the presence of another station's transmission while transmitting (*collision detect*).
- A total worst-case round trip signal propagation delay (including actual propagation time, synchronization time for all intervening electronics, and signal rise time degradation) of less than 464 bit times (equal to 46.4 μs for this 10 Mbps channel).

7.1.3 Functions Provided by the Channel

The channel hardware provides the following functions in the performance of its role:

- Means for transmitting and receiving serial bit streams between the data link layer and the media.
- Generation of clock for synchronization and timing.
- Means for detecting carrier (non-idle channel).
- Means for detecting collisions (simultaneous transmission attempts by multiple stations).
- Coding and decoding of the data link bit stream into a self-synchronizable sequence of electrical signals suitable for transmission on the media provided by the channel.
- Generation and removal of coding-specific preamble information (a synchronizing header sequence inserted before the first bit of the frame) to ensure that all channel electronics are brought to a known steady-state before the data link frame is transmitted.

7.1.4 Implementation of the Channel

The physical channel specification is implementation dependent; most of the channel hardware is fully specified, and little leeway is given to the individual designer. This is done in the interest of compatibility; any system which allows different implementers to use different channel cables, connectors, clock speeds, and the like will not be compatible across manufacturer boundaries. Only the design of channel components which are not critical to system compatibility is left to the implementer.

7.1.4.1 General Overview of Channel Hardware

The channel minimally consists of the following functional blocks:

- Passive broadcast medium (coaxial cable).
- Transceiver (transmitter-receiver for the coaxial cable).
- Means for connecting transceivers to a coaxial cable segment and for connecting coaxial cable segments together.
- Channel clock.
- Channel data encoder and decoder.
- Preamble generator and remover.
- Carrier and collision detect circuits.

The coaxial medium is the only element common to the entire network. A transceiver is required for connection to the medium, and must be located adjacent to the coaxial cable. The last four components are generally located within, and tightly coupled to, the station hardware implementing the data link function.

It may be desirable to physically separate the transceiver from the rest of the channel hardware. This permits topological flexibility, packaging advantages, and improved system availability, and allows for independent manufacture of station hardware and transceivers. To ensure that compatibility is maintained, a physical interface (known as the *transceiver cable*) is identified and specified to connect the transceiver to the station.

Finally, it is necessary to add *repeaters* to the system to reach the maximum allowable distance between stations, and to provide additional topological flexibility. Repeaters are implemented using standard transceivers, plus a finite state machine.

7.1.4.2 Compatibility Interfaces

There are a number of possibilities for implementing systems or subsystems compatible in whole or in part with this specification. It is important that all implementations be compatible at some point, so that heterogeneous systems from different manufacturers' implementations can be interconnected on the same medium. It is not necessary in every case to implement all of the components described here; e.g., it is possible to design an integrated station/transceiver (without requiring the transceiver cable). The implementer must make the required tradeoffs between topological flexibility, system availability, configurability, user needs, and cost, when designing the system.

For a device to be considered compatible, it must meet the applicable requirements at either the transceiver cable or the coaxial cable interface, as appropriate, in addition to the Data Link compatibility required for all stations connected to the network.

All Ethernet implementations must be compatible at the coaxial cable.

If a transceiver cable is used, it should be the one specified in this document. This allows device manufacturers to build hardware compatible with the Ethernet at the transceiver cable level, without concerning themselves with the details of transceiver implementation. Devices implementing transceiver cable compatibility should be capable of using transceivers designed and built by another manufacturer, on the specified coaxial cable.

Equipment designed for connection to the specified coaxial cable either without a physically separate transceiver or with a non-standard transceiver cable interface will be capable of communication. However, a sacrifice may have been made with respect to interchangeability with other stations.

This scheme of multiple compatibility interfaces allows individual designers some flexibility in making system tradeoffs, yet allows cable manufacturers, transceiver manufacturers, and systems manufacturers to use standard commodity parts to produce a compatible communications system.

7.1.5 Channel Configuration Model

Certain physical limits have been placed on the physical channel. These revolve mostly around maximum cable lengths (or maximum propagation times), as these affect the slot time as defined in the data link. While the precise specification (in later sections) states these maxima in terms of propagation times, they were derived from the physical configuration model described here.

- A coaxial cable, terminated in its specified characteristic impedance at each end. This constitutes a cable segment and may contain a maximum of 500 meters of coaxial cable.

- A maximum of 2 repeaters in the path between any two stations. Repeaters do

not have to be located at the ends of segments, nor is the user limited to one repeater per segment. In fact, repeaters can be used not only to extend the length of the channel, but to extend the topology from one to three-dimensional. A repeater may be constructed from two "half-repeaters" separated by a point-to-point link. Thus, a maximum of 4 such half-repeaters can be located in the path between any two stations in the network. Repeaters occupy transceiver positions on each cable segment. Each repeater and each station count towards the maximum number of transceivers on a segment.

- A maximum multidrop (Ethernet) coaxial cable length along the longest path between any two transceivers of 1500 meters. The propagation velocity of the coaxial cable is assumed to be 0.77 c worst-case. (c is the velocity of light *in vacuo*; 300,000 kilometers per second.) The total round-trip delay for all the multidrop (Ethernet) coaxial cable in the system is therefore 13 μs worst-case.

- A maximum of 50 meters of transceiver cable between any station and its associated transceiver. Note that in the worst case the signal must pass through six 50-meter transceiver cables, one at the transmitting station, one at the receiving station, and 2 at each repeater (two repeaters possible). The propagation velocity of the transceiver cable is assumed to be .65 c worst-case. The total round-trip delay for all the transceiver cables is therefore 3.08 μs worst-case.

- A maximum aggregate of 1000 meters of point-to-point links between any two stations in the system. These point-to-point links are normally integrated within repeaters, and can be used to connect cable segments in different buildings (see Fig. 7-1c). However, this does not mean that there can be only one point-to-point link in the system. There can be as many point-to-point links as repeaters (though not all can be 1000 meters). The worst-case propagation velocity of the point-to-point link cable is assumed to be .65 c; the round-trip propagation delay for 1000 meters is 10.26 μs.

Table 7-1 summarizes the allocation of the round-trip propagation delay to the individual components in the channel. Figure 7-1 shows a minimum, typical, and large-scale channel configuration.

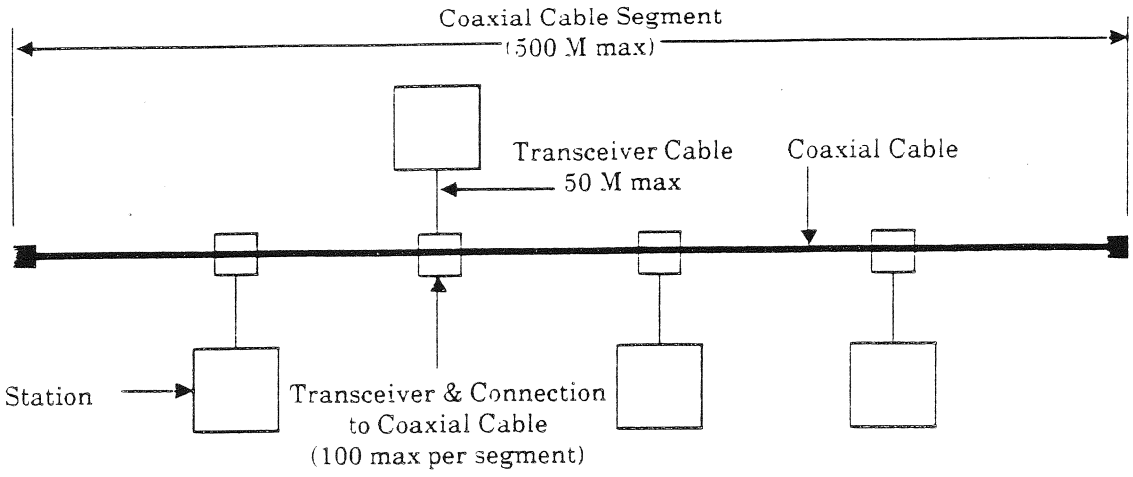


Figure 7-1a: Minimal Configuration

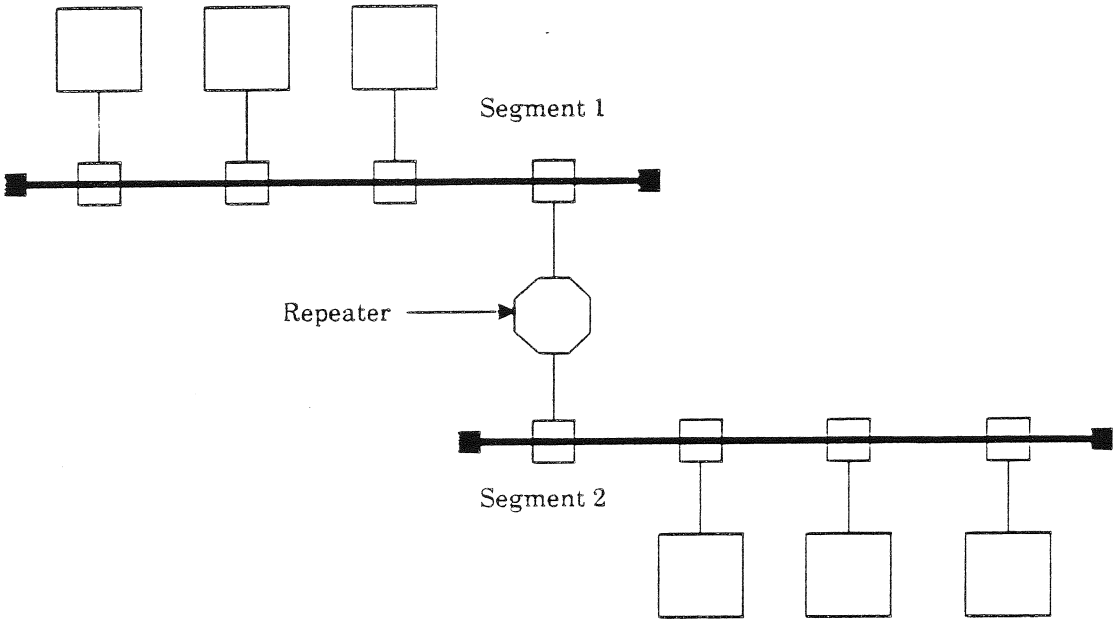


Figure 7-1b: A Typical Medium-scale Configuration

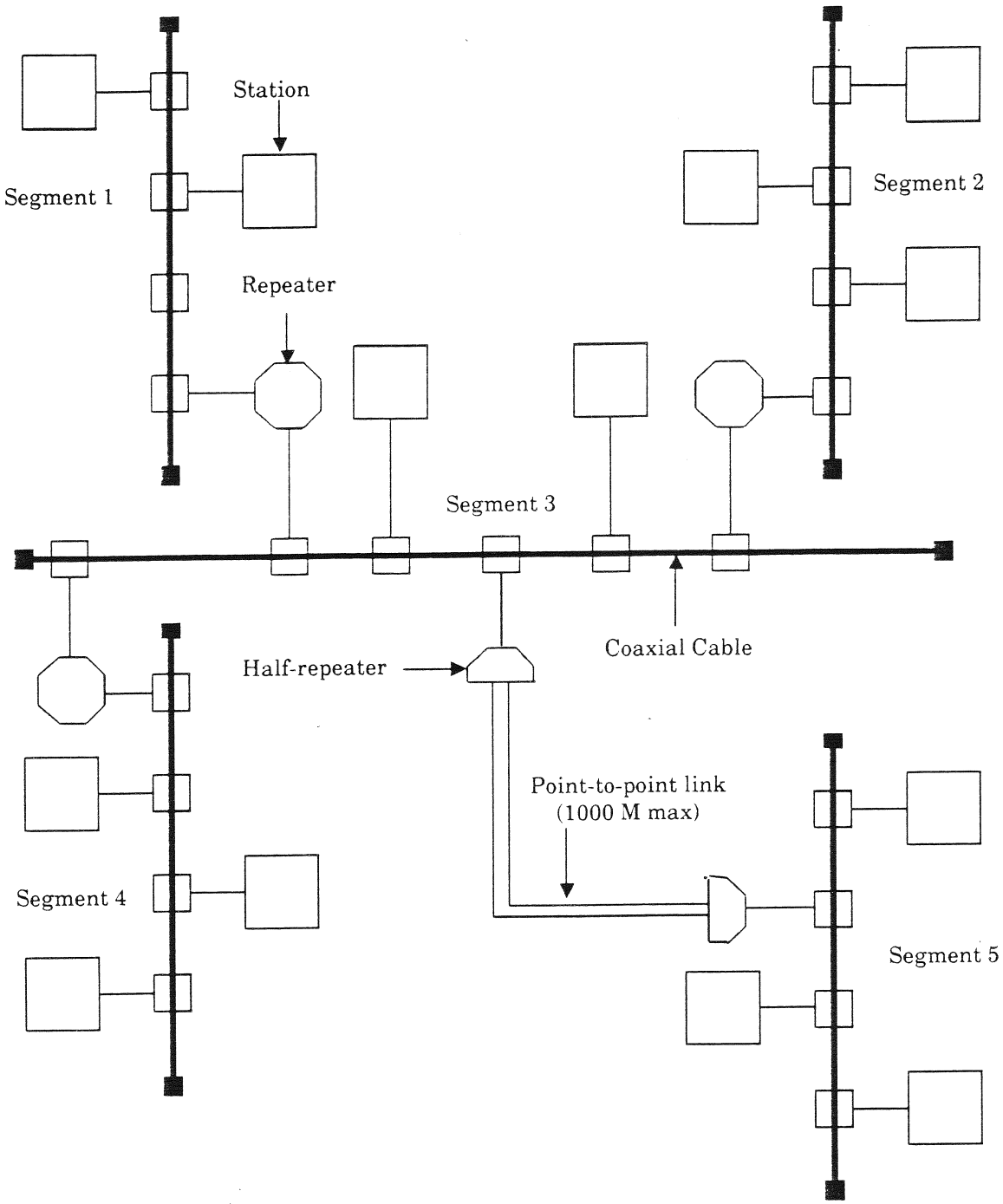


Figure 7-1c: A TypicalLarge-scale Configuration

Table 7-1: Physical Channel Propagation Delay Budget (Note 1)

Element	Unit Steady-State Delay	Unit Startup Delay	# Units Forward Path (Note 2)	# Units Return Path	Total Delay
Encoder	0.1 μs	0.1 μs	5	5	2.0 μs
Transceiver Cable	5.13 ns/M	0	300 M	300 M	3.08 μs
Transceiver (transmit path)	0.05 μs	0.30 μs	3	3	2.10 μs
Transceiver (receive path)	0.05 μs	0.60 μs	3	0	1.95μs
Transceiver (collision path)	0	0.9 μs	0	3	2.70 μs
Coaxial Cable	4.33 ns/M	0	1500 M	1500 M	12.99 μs
Point-to-Point Link Cable	5.13 ns/M	0	1000 M	1000 M	10.26 μs
Point-to-Point Link Driver	0.10 μs	0	2	2	0.40 μs
Point-to-Point Link Receiver	0.10 μs	0	2	2	0.40 μs
Repeater (repeat path)	0.2 μs	0.4 μs	2	0	1.20 μs
Repeater (collision path)	0.2 μs	0.2 μs	0	2	0.80 μs
Carrier Sense	0	0.2 μs	5	0	1.0 μs
Collision Detect	0	0.2 μs	0	5	1.0 μs
Signal Rise Time (to 70% in 500 M) (Note 3)	0	0.1 μs	3	0	0.30 μs
Signal Rise Time (50% to 94% in 500 M) (Note 4)	0	2.0 μs	0	3	6.0 μs
Collision Fragment Timer Tolerance	0	0.2 μs	0	1	0.2 μs
		Total Worst-Case Round-Trip Delay			46.38 μs

Note 1: All quantities given are worst-case (both number of units and unit delays per unit).

Note 2: The propagation delay has been separated into "forward-path" and "return path" delay. This is because in one direction it is *carrier sense* which is being propagated through the channel, and in the return direction it is *collision detect* which is being propagated. The two signals have different propagation delays.

Note 3: In the worst case, the propagated signal must reach 70% of its final value to be detected as valid carrier at the end of 500 meters of coaxial cable. This rise time must be included in the propagation delay budget.

Note 4: In the worst case the propagated collision on the return path must reach 94% of its final value to be detected as a collision at the end of 500 meters of coaxial cable.

7.1.6 Channel Interfaces

The channel specification hinges around three well-defined entities; the transceiver and coaxial cables (shown as compatibility interfaces in Figure 4-1), and the logical interface between the physical channel and the data link controller (shown in Figure 4-4). Note that the first two are physical interfaces specific to the channel, and are specified in the interest of compatibility. The last is provided as a means by which the data link controller can interact with the physical channel.

The channel access component of the logical interface (discussed in 4.4.1) comprises the collision and carrier detect functions described in 7.5.2 and 7.5.3, as well as the actual transmission of signals on the media. The data encoding and decoding functions described in 4.4.1 comprise the generation and decomposition of encoded signals suitable for transmission (described in 7.5.1), the generation and removal of code-specific preamble (described in 7.5.1.3 and 7.5.4.1), and the serial bit stream interface between the layers.

Section 5 describes the interface between the Data Link and Physical Layers as a collection of Pascal procedures, functions, and shared variables. The specifications in Section 6 show how peer Data Link Layers use this interface to communicate. No attempt will be made to model the operation of the physical channel in Pascal.

The interface between the Data Link and Physical Layers is supported by the physical channel hardware which provides the following services: the ability to send and receive bit streams, physical channel timing, *carrier sense*, and *collision detect* signaling to the data link.

The remainder of this section specifies the requirements for compatibility at both the transceiver cable and the coaxial cable. In addition, the specification for the transceiver, which interfaces the transceiver cable to the coaxial cable is given, as well as the specification for the logic required between the transceiver cable and the interface to the data link.

7.2 Transceiver Cable Compatibility Interface Specifications

The transceiver cable is the means by which a physically separate transceiver is connected to a station. It provides one of the compatibility interfaces described in 7.1.4.2.

7.2.1 Transceiver Cable Signals

The transceiver cable carries power and three signals: Transmit, Receive, and Collision Presence. Each signal is carried on a twisted pair of conductors in the cable. These twisted pairs are referred to as *power pair* and *signal pairs* where appropriate below.

7.2.1.1 Transmit Signal

The transmit pair carries encoded data for which the data link is requesting transmission on the channel. This signal is generated by the data encoder, with the transceiver cable drive characteristics specified in 7.2.4.

7.2.1.2 Receive Signal

The receive pair carries encoded data from the transceiver to the station. It typically goes to the data decoder and the carrier sense circuitry. In the steady-state, all transitions and lack of transitions on the coaxial cable become transitions and lack of transitions on the receive pair, with the transceiver cable drive characteristics specified in 7.2.4. (During start-up, the first few bits may be absorbed by the transceiver to attain steady-state.)

In the case of a station transmitting without collision interference, the station's own transmit transitions on the coaxial cable will also appear on the receive pair, after a delay due to propagation through the transceiver. During collisions (whether or not that transceiver is involved in the collision) transitions on the receive lead are undefined; they may or may not meet decoder phase requirements, or they may not be present at all for extended periods. Thus, the receive signal on the transceiver cable cannot be used alone to deterministically generate the carrier sense signal. This is described in more detail in 7.5.3.

7.2.1.3 Collision Presence Signal

The collision presence pair is used by the transceiver to indicate the presence of multiple transmission attempts on the coaxial cable. This is done by transmitting a signal with a 10MHz fundamental frequency through the standard transceiver cable driver (described in 7.2.4). An oscillator is used instead of a simple level shift to allow AC coupling at the transceiver. Transceivers must activate the collision presence signal in both of the following conditions: 1) there are two stations transmitting simultaneously and the transceiver in question is connected to one of those stations, 2) there are three or more stations transmitting simultaneously (independent of whether the transceiver in question is connected to one of the transmitting stations).

7.2.1.4 Power

A pair of wires is designated for providing power to the transceiver. When the transceiver cable is implemented, the station end of the cable must supply a voltage between +12 and +15 Vdc ± 5% (open circuit) with at least 0.5 Amperes available to the cable for remotely powering the transceiver. The power source must meet applicable requirements for NFPA Class 2 wiring devices, per NEC Article 725, reference [12].

7.2.2 Transceiver Cable Parameters

7.2.2.1 Mechanical Configuration

The transceiver cable consists of four stranded, shielded twisted pair conductors, plus an overall shield and insulating jacket. The conductor and jacket insulating material may be polyethylene or other suitable material. The flammability characteristics of the insulating material must be suitable for the installed environment.

7.2.2.2 Characteristic Impedance

The differential mode characteristic impedance of all signal pairs shall be 78 Ω, ± 5 Ω, in the configuration.

7.2.2.3 Attenuation

The attenuation of any signal pair shall not exceed 3.0 dB (measured at 10 MHz) nor 2.1 dB measured at 5 MHz for the total length between the transceiver and the station.

7.2.2.4 Velocity of Propagation

The minimum velocity of propagation of any signal pair shall be 0.65 c.

7.2.2.5 Timing Distortion

Timing distortion of any signal pair shall not exceed ± 1 ns at the end of 50 meters of cable when driven with random 10 Mbps data encoded in accordance with 7.5.1.

7.2.2.6 Resistance

The resistance of the conductors used for the power pair shall not exceed 40 mΩ/meter. For a 50 meter transceiver cable, this implies that the distribution impedance of the power supplied to the transceiver shall not exceed 4 Ω.

7.2.2.7 Transfer Impedance

The common mode transfer impedance of the transceiver cable shall not exceed the values shown in Figure 7-2 as a function of frequency. The differential mode transfer impedance of the cable with respect to any signal pair shall be 20 dB lower than the specified common mode transfer impedance.

7.2.3 Transceiver Cable Connectors

The connectors used at the ends of the transceiver cable shall be 15 conductor 'D' subminiature types (MIL-C-24308 or commercial equivalent). The end of the cable that mates with the transceiver must use a female connector with a slide latch

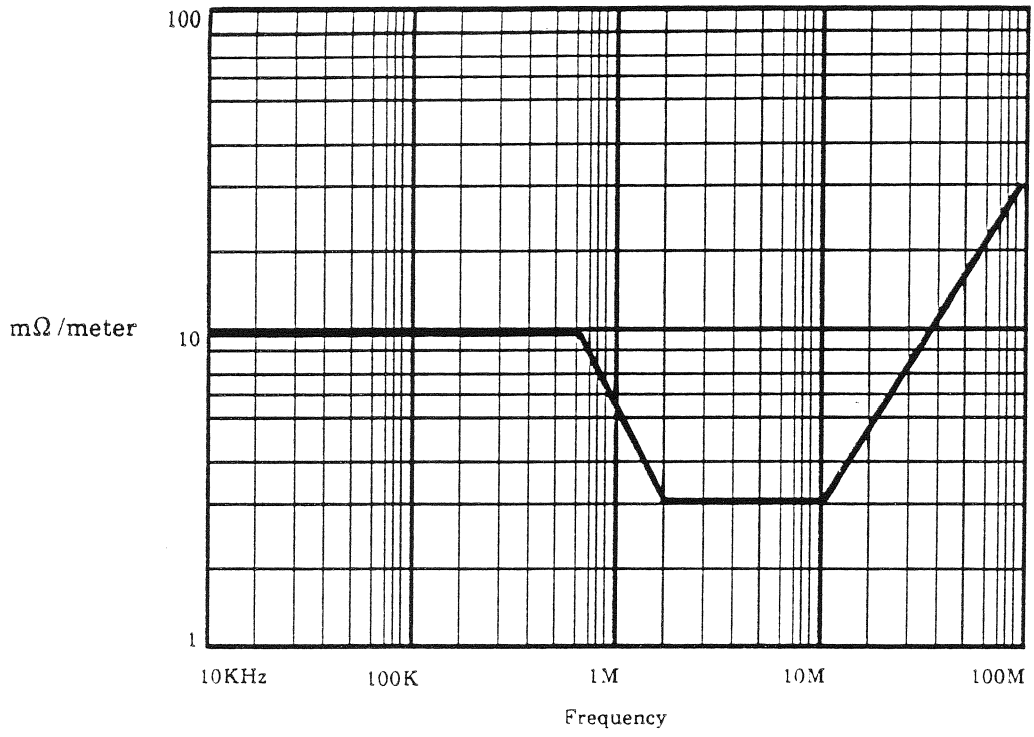


Figure 7-2: Maximum Transceiver Cable Transfer Impedance

assembly. The transceiver must provide a mating male connector with locking posts. The other end of the transceiver cable (which mates with a female connector at the station) must use a male connector with locking posts. The station must provide a female connector with the slide latch assembly.

Because of the end-to-end matching of the connectors, transceiver cables may be extended by concatenating transceiver cable sections. (The transceiver cable sections function as extension cords.) A cable with multiple sections must still meet the cable characteristics of 7.2.2.

The pin assignment is given in the following table:

Transceiver Cable Connector Pin Assignment

1. Shield (See note)
2. Collision Presence +
3. Transmit +
4. Reserved
5. Receive +
6. Power return
7. Reserved
8. Reserved
9. Collision Presence -
10. Transmit -
11. Reserved
12. Receive -
13. Power
14. Reserved
15. Reserved

Note: Shield must be terminated to connector shell as well as pin 1, and must not

be connected to any other pin. No connections are allowed on the reserved pins.

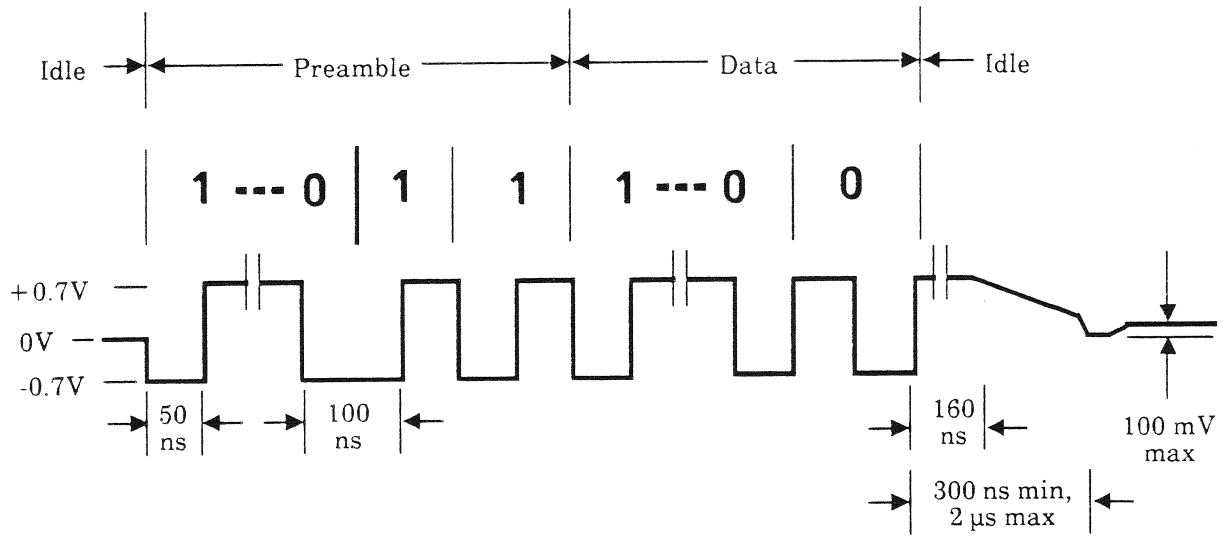
Metal, metallized plastic, or otherwise shielded connector backshells must be used to ensure shield integrity.

7.2.4 Transceiver Cable Drive

This section describes the requirements for driving any of the signal pairs in the transceiver cable: transmit, receive, and collision presence.

The AC signal levels presented to the transceiver cable shall be ± 700 mV nominal (± 550 mV min, ± 1200 mV max), balanced differential drive into 78 ± 5 Ω. Signal waveform shall be as shown in Figure 7-3.

The DC common mode voltage on the transceiver cable shall be determined by the controller (not the transceiver) and shall be in the range of 0 to 5 volts.



1. Voltages are nominal, measured differentially at output of transceiver cable driver.
2. Rise and fall times meet 10,000 series ECL requirements.

Figure 7-3: Typical Transceiver Cable Waveform

The transceiver cable driver must be capable of maintaining the specified minimum differential signal into the worst case low cable impedance (73 Ω differential, 18.5 Ω common mode) in the environment specified in Section 7.7.

The idle state of the output shall be nominally 0 volts differential; the first transition presented must be negative-going, the last transition must be positive-going. The transceiver cable driver must return to the idle state (0 volts) no less than 300 ns nor less than 2 µs following the last positive-going transition of the frame. During this 'return-to-idle' time, the magnitude of the voltage presented to the transceiver cable by the driver shall be not greater than 100 mV differentially.

The transceiver cable drive circuits must be capable of driving the isolation transformer located within the transceiver and specified in 7.4.3.

A typical transceiver cable drive circuit is given in Appendix D.

7.2.5 Transceiver Cable Receive

The following sections specify the requirements for receiving signals from any signal pair in the transceiver cable: transmit, receive, and collision presence. The circuit must be capable of receiving the signals from the transceiver cable driver specified in 7.2.4 through the cable specified in 7.2.2 in the worst case. A typical receive circuit is given in Appendix D.

7.2.5.1 Load Impedance and Termination

The termination impedance shall be 78 Ω ± 5% differential mode, and 18.5 Ω minimum AC common-mode, with less than 2% common-mode imbalance, over the frequency range of 3-20 MHz.

7.2.5.2 Common Mode and CMRR

The common mode range and the common mode rejection ratio shall be sufficient to maintain a 5:1 signal-to-noise ratio in the environment specified in 7.7, measured at the input to the transceiver cable receiver.

The DC common-mode voltage on the transceiver cable shall be determined by the controller (not the transceiver) and shall be in the range of 0 to 5 volts.

7.2.5.3 Transceiver Cable Squelch

The transceiver cable receiver shall provide squelch circuitry to ensure that adequate noise immunity exists. This squelch circuit must ensure that the transceiver cable receiver is not enabled unless the magnitude of the received signal from the transceiver cable exceeds 175 mV, minimum.

7.3 Coaxial Cable Compatibility Interface Specifications

The coaxial cable is the common, shared broadcast medium through which stations communicate. It provides one of the compatibility interface points described in 7.1.4.2.

7.3.1 Coaxial Cable Component Specifications

The cable is of constant impedance, coaxial construction. It is terminated at each end by a *terminator* (specified in 7.3.1.3), and connection is provided for each transceiver. Coaxial cable connectors are used to make the connection from the cable to the terminators, and between cable sections (if needed). The cable has various electrical and mechanical requirements which must be met to ensure proper operation.

7.3.1.1 Coaxial Cable Parameters

7.3.1.1.1 Characteristic Impedance

The average characteristic impedance of the cable shall be 50 $\pm$ 2 Ω, measured according to MIL-STD C17-E. Periodic variations in impedance along a single piece of cable may be up to $\pm$ 3 Ω sinusoidal, centered around the average value, with a period $<$ 2 meters. Note that the proper operation of the network is dependent upon the cable characteristic impedance; its value and tolerance are critical.

7.3.1.1.2 Attenuation

The attenuation of a cable segment shall not exceed 8.5 dB measured at 10 MHz, nor 6.0 dB measured at 5 MHz. If the means of connecting any transceivers on the segment requires the severing of the cable as specified in 7.3.1.4, then the attenuation of such transceivers must also be included in the segment attenuation.

7.3.1.1.3 Velocity of Propagation

The minimum acceptable velocity of propagation is 0.77 c.

7.3.1.1.4 Mechanical Requirements

The cable used should be suitable for routing in various environments, including but not limited to, dropped ceilings, raised floors, and cable troughs. The jacket must provide insulation between the cable sheath and any building structural metal. Also, the cable must be capable of accepting coaxial cable connectors, described in 7.3.1.2. The cable must also conform to the following requirements:

- Center conductor must be 0.0855" $\pm$ .0005" diameter, solid copper.

- Core dielectric material must be foamed.
- Inside diameter of the innermost shield must be .242" minimum.
- Outermost shield must be greater than 90% coverage tinned copper braid, with an O.D. of 0.326" ± .007"
- Jacket O.D. must be 0.365" minimum, 0.415" maximum.
- Cable concentricity must be such that the center of the center conductor is within 0.020" of its ideal concentric position with respect to the jacket.

The cable must also meet applicable flammability criteria and local codes for the installed environment. Different (e.g., polyethylene and FEP dielectric) types of cable sections may be interconnected, while meeting the sectioning requirements of 7.6.1

7.3.1.1.5 Timing Distortion

Timing distortion shall not exceed ± 7 ns at the end of 500 meters of cable when driven with random 10 Mbps data encoded in accordance with 7.5.1.

7.3.1.1.6 Jacket Marking

The cable jacket must be marked with annular rings in a color contrasting with the background color of the jacket. The rings must be spaced at 2.5 meters ± 5 cm from the previous ring, regularly along the entire length of the cable. It is permissible for the 2.5 meter spacing to be interrupted at discontinuities between cable sections joined by connectors. (See 7.6.2 for transceiver placement rules which mandate cable markings.)

7.3.1.1.7 Transfer Impedance

The transfer impedance of the cable shall not exceed the values shown in Figure 7-4 as a function of frequency.

7.3.1.1.8 DC Resistance

The loop resistance of a coaxial cable segment shall not exceed 5 Ω, including any connectors in the loop. If the means of connecting any transceivers on the segment requires the severing of the cable as specified in 7.3.1.4, then the connectors plus all internal resistance in the shield and center conductor within such transceivers must also be included in the loop resistance measurement.

7.3.1.2 Coaxial Cable Connectors

Coaxial cable connectors are used to join cable sections and to attach terminators. Three types of connectors may be necessary; male plugs, female jacks, and female-

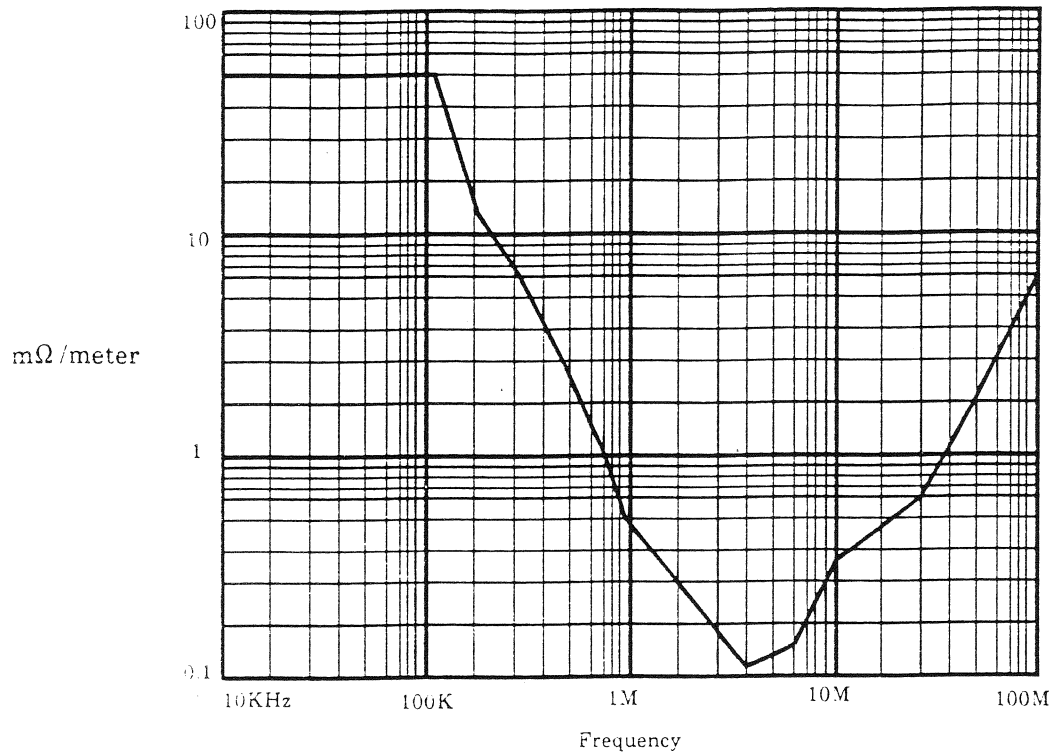


Figure 7-4: Maximum Coaxial Cable Transfer Impedance

to-female barrels. Plugs are used exclusively at the ends of all cable sections. Jacks are used to house cable terminators. Barrels are used to join cable sections.

All connectors are N series, 50 Ω constant impedance types. Since the frequencies present in the transmitted data are well below UHF range (being band-limited to approximately 20 MHz), military versions of the connectors are not required (but are acceptable).

Means must be provided to ensure that the connector shell (which connects to the cable sheath) does not make contact with any building metal, or other unintended conductor.

7.3.1.3 Coaxial Cable Terminators

Coaxial cable terminators are used to provide a termination impedance for the cable equal in value to its characteristic impedance, thereby eliminating any reflection from the ends of the cables. Terminators shall be packaged within an inline female jack connector. The termination impedance shall be 49.9 Ω ± 1% measured from 0-20 MHz, with the magnitude of the phase angle of the impedance not to exceed 5 degrees. The terminator power rating shall be 1 watt or greater.

7.3.1.4 Transceiver-to-Coaxial Cable Connections

A means must be provided to allow for attaching a transceiver to the coaxial cable. The connection must disturb the transmission line characteristics of the cable as little as possible; it must present a predictably low shunt capacitance, and therefore a negligibly short stub length. For this reason, the transceiver must be located as close to its cable connection as possible; they are normally considered to be one assembly. Long (greater than 3 cm) connections between the coaxial cable and the input of the transceiver are not acceptable.

The transceiver-to-coaxial cable connection shall present less than 2 picofarads shunt capacitance to the coaxial cable, not including any transceiver electronics.

If the design of the connection is such that the coaxial cable must be severed to install the transceiver, the coaxial cable segment must still meet the sectioning requirements of 7.6.1. Any coaxial connectors used on a severed cable must be type N, as specified in 7.3.1.2. The loop resistance of such a transceiver shall not exceed 25 mΩ, including all connectors and series resistance in the shield and center conductor within the transceiver.

Note that the loop resistance allowance for a single transceiver-to-coaxial cable connection may cause the segment loop resistance constraint of 7.3.1.1.8 to be a more severe limitation on the maximum number of transceivers permitted on a single segment than that specified in 7.6.2.

7.3.2 Coaxial Cable Signaling

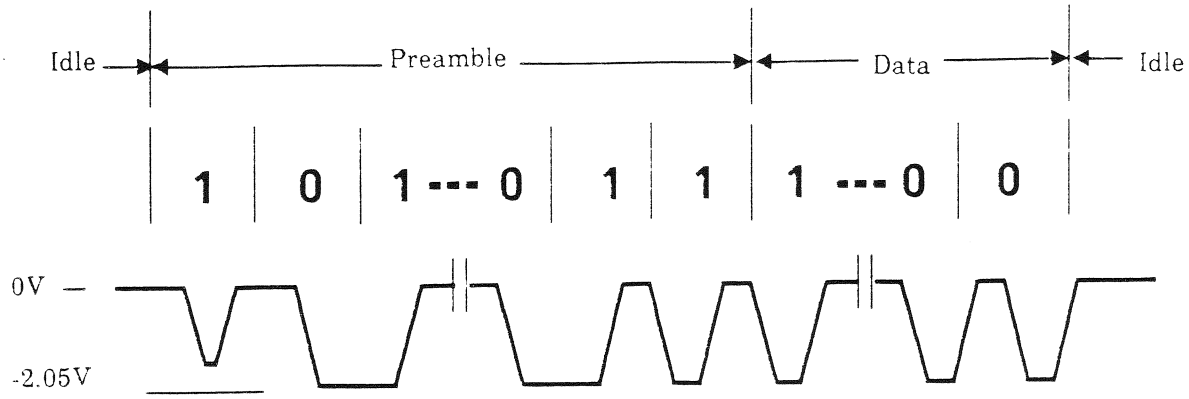
The AC component of the signal on the coaxial cable due to a single transceiver as measured on the coaxial cable immediately adjacent to the transceiver connection shall be between ± 14 mA and I_{DC}, where I_{DC} is the average DC component of the signal. The DC component of the signal (I_{DC}, including effects of timing distortion) shall be 20.5 mA nominal (18 mA minimum, 24 mA maximum). The actual current measured at a given point on the cable is a function of the transmitted current and the cable loss to the point of measurement. Positive current is defined as current out of the center conductor of the cable (into the transceiver). Cable loss is specified in 7.3.1.1.2.

The 10%-90% rise and fall times shall be 25 ± 5 ns. Rise and fall times must be matched to within 1 ns (at the coaxial cable driver). Figure 7-5 shows typical waveforms present on the cable. Harmonic content generated from a 10 MHz fundamental periodic input shall meet the following requirements:

Second and Third Harmonics:	no less than 20 dB down from fundamental
Fourth and Fifth Harmonics:	no less than 30 dB down from fundamental
Sixth and Seventh Harmonics:	no less than 40 dB down from fundamental
All Higher Harmonics:	no less than 50 dB down from fundamental

The signals as generated from the encoder (described in 7.5.1.1) shall appear on the coaxial cable without any inversions.

The transceiver shall be able to produce its specified output current onto the coaxial cable with at least one other transceiver transmitting simultaneously.



1. Voltages given are nominal; worst case is given in text.

2. Rise time is 25 ns nominal..

3. Voltages are measured on coaxial cable adjacent to transceiver.

Figure 7-5: Typical Coaxial Cable Waveform

7.4 Transceiver Specifications

7.4.1 Transceiver-to-Coaxial Cable Interface

The following sections describe the interface between the transceiver and the coaxial cable. Positive current is defined as current into the transceiver (out of the center conductor of the cable).

7.4.1.1 Input Impedance

The shunt capacitance presented to the coaxial cable by the transceiver circuitry (not including the means of attachment to the coaxial cable) shall not exceed 2 picofarads. The shunt resistance presented to the coaxial cable shall be greater than 100 KΩ.

These conditions must be met in both the power off and the power on, not transmitting states.

7.4.1.2 Bias Current

The transceiver must draw between -2 and +25 μA in the power-off and the power-on, not transmitting states.

7.4.1.3 Transmit Output Levels

Signals received from the transceiver cable transmit pair must be transmitted onto the coaxial cable with the characteristics specified in 7.3.2. Note that 7.3.2 specifies the current level on the coaxial cable. Since the coaxial cable proceeds in two directions away from the transceiver, the current into the transceiver is actually twice the current measured on the coaxial cable.

Timing distortion shall not exceed +2 ns for a 50%/50% duty cycle input on the transceiver cable transmit pair.

7.4.2 Transceiver-to-Transceiver Cable Interface

7.4.2.1 Transmit Pair

The transceiver must present the transceiver cable receive characteristics specified in 7.2.5 to the transmit pair. At the start of a frame transmission, no more than 2 bits (two 100 ns bit cells) of information may be received from the transmit pair and not transmitted onto the coaxial cable. In addition, it is permissible for the first bit sent onto the coaxial cable (the third bit from the transmit pair, worst case) to contain phase violations or invalid data. However, all successive bits of the frame shall be valid and meet encoding rules. The fourth bit seen on the transmit pair must be guaranteed valid on the coaxial cable.

The transceiver shall provide transceiver cable receive squelch as specified in 7.2.5.3, for the transmit pair.

The steady-state propagation delay between the transmit pair input and the coaxial cable output shall not exceed 50 ns. There is no signal inversion between the transceiver cable transmit pair and the coaxial cable.

7.4.2.2 Receive Pair

The transceiver must present the transceiver cable drive characteristics specified in 7.2.4 to the receive pair.

The signal from the coaxial cable shall pass through AC coupling with an appropriate time constant before proceeding to the receive pair. The time constant should compensate for the coaxial cable timing distortion.

At the start of a frame reception from the coaxial cable, no more than 5 bits (five 100 ns bit cells) of information may be received from the coaxial cable and not transmitted onto the receive pair. In addition, it is permissible for the first bit sent over the receive pair (the sixth bit from the coaxial cable, worst case) to contain encoding phase violations or invalid data. However, all successive bits of the frame shall be valid and meet encoding rules. The seventh bit seen on the coaxial cable must be guaranteed valid on the receive pair.

Timing distortion on the receive pair shall not exceed ±2 ns with a ±200 mV peak sinusoidal input from the coaxial cable.

Following the last positive-going transition at the end of a frame on the receive pair, the transceiver must guarantee that no additional transitions are generated on the receive pair for a period of 160 ns, in the environment specified in 7.7.

The steady-state propagation delay between the coaxial cable and the receive pair output shall not exceed 50 ns. There is no signal inversion between the coaxial cable and the transceiver cable receive pair.

The transceiver shall not enable transmissions onto the receive signal pair until the DC signal on the coaxial cable exceeds -300 mV, minimum.

7.4.2.3 Collision Presence Pair

The transceiver must present the transceiver cable drive characteristics specified in 7.2.4 to the collision presence pair. The signal presented to the collision presence pair shall be a periodic waveform with a 10 MHz ± 15% fundamental frequency. The duty cycle of the signal presented to the collision pair shall be no worse than 60%/40%-40%/60%. This signal shall be presented to the collision presence pair no more than 9 bit times (900 ns) after the average DC signal on the coaxial cable implies a simultaneous transmission attempt by more than one station (a collision), in the worst case.

7.4.2.4 Power Pair

The transceiver cable provides power which may be used for operation of the transceiver electronics. The power available shall be as described in 7.2.1.4. The loop resistance of the transceiver cable is 4 Ω maximum, for a 50 meter cable with the resistance specified in 7.2.2.6. In order for the transceiver to derive its operating power from the power pair, circuitry must be employed to provide the required electrical isolation specified in 7.4.3.

7.4.2.5 Transceiver-to-Transceiver Cable Connections

Transceivers shall connect to the signal pairs of the transceiver cable through a coupling transformer. The magnetizing inductance of this transformer shall be greater than 27 μH, seen from the transceiver cable. This transformer shall provide the electrical isolation required in 7.4.3. Note that this transformer results in a low resistance DC connection differentially, across each of the transceiver cable signal pairs. Transceiver cable drive and receive circuitry at both ends of the cable must take this into account.

7.4.3 Electrical Isolation

The transceiver must provide electrical isolation between all conductors of the

transceiver cable and both conductors of the coaxial cable. The isolation impedance shall be greater than 250 KΩ, measured between any conductor (including shield) of the transceiver cable and either the center conductor or shield of the coaxial cable, at 60 Hz. The breakdown voltage of the isolation means provided shall be at least 270 VAC, rms.

Specific market requirements may mandate greater isolation voltages than are required by this specification.

7.4.4 Reliability

No single, nor, as a design goal, double component failure within the transceiver electronics shall impede communication among other transceivers on the coaxial cable. Connectors and other passive components comprising the means of connecting the transceiver to the coaxial cable shall be designed to minimize the probability of total network failure.

A transceiver may not generate spurious or unspecified signals onto the coaxial cable under conditions where the voltage applied to the power pair of the transceiver cable is less than specified in 7.2.1.4, or the loop resistance of the power supplied is greater than 4 Ω. The transceiver is not required to operate properly with respect to its attached controller under these conditions.

7.4.5 Common Mode Current Shunt

The transceiver shall provide a shunt for high frequency, common mode currents between the transceiver cable sheath and the coaxial cable sheath. The impedance of this shunt shall be less than 10 Ω between 3 MHz and 30 MHz, and greater than 250 KΩ at 60 Hz.

7.4.6 Transmit Watchdog Timer

Transceivers shall inhibit continuous transmissions onto the coaxial cable of duration greater than 150 μs, but shall not inhibit continuous transmissions of duration less than 20 μs. Once the cause of the continuous transmission has been removed, transmissions shall be re-enabled within at most 100 μs.

7.4.7 Collision Presence Test

Transceivers shall implement a self-test to indicate to the controller the operational status of the collision presence circuitry in the transceiver. The transceiver shall indicate the functioning of the collision presence circuitry by transmitting a signal as specified in 7.2.1.3 onto the collision presence pair of the transceiver cable following each transmission (on the transmit pair of the transceiver cable) from its associated controller. The signal shall begin no less than 360 ns following the last positive-going transition on the transmit pair. The signal shall be sent for a nominal 500 ns (300 ns minimum). The signal shall end

within 2 μs following the last transition on the transmit pair.

The method used to stimulate the collision presence signal within the transceiver during this test is implementation-dependent. However, care should be taken to exercise as much of the collision presence circuitry as possible. Under no circumstances shall this collision presence test cause spurious or unspecified transmissions to occur on the coaxial cable at any time.

7.5 Channel Logic

The following sections describe the functions that must be performed to properly interface between the data link and the transceiver cable. They are normally implemented as logic, typically within the same device implementing the data link layer.

7.5.1 Channel Encoding

The channel shall use Manchester phase encoding, with a data rate of 10 Mbps, ± .01%, measured at the encoder clock. Thus, each bit cell is 100 ns long.

The following section describes the requirements for encoding and decoding signals to be transmitted on, or received from the coaxial or transceiver cables.

7.5.1.1 Encoder

The encoder is used to translate physically separate signals of clock (synchronization) and data into a single, self-synchronizable serial bit stream, suitable for transmission on the coaxial cable by the transceiver.

During the first half of the bit cell time, the serial signal transmitted is the logical complement of the bit value being encoded during that cell. During the second half of the bit cell time, the uncomplemented value of the bit being encoded is transmitted. Therefore, there is always a signal transition (either positive-going or negative-going, depending on the bit being encoded) in the center of each bit cell. A timing diagram for a typical bit stream is given in Figure 7-6.

The encoder output drives the transmit pair of the transceiver cable, and ultimately, the coaxial cable through the transceiver. The encoder output timing distortion must not exceed 0.5 ns. The encoder shall provide the defined output for the first (and all subsequent) bits presented to its input. All information submitted for encoding shall appear at the output of the encoder.

The steady-state propagation delay of the encoder shall not exceed 100 ns. The delay between the presentation of valid clock and data to the encoder and the presentation of the encoded signal to the transmit pair shall not exceed 100 ns.

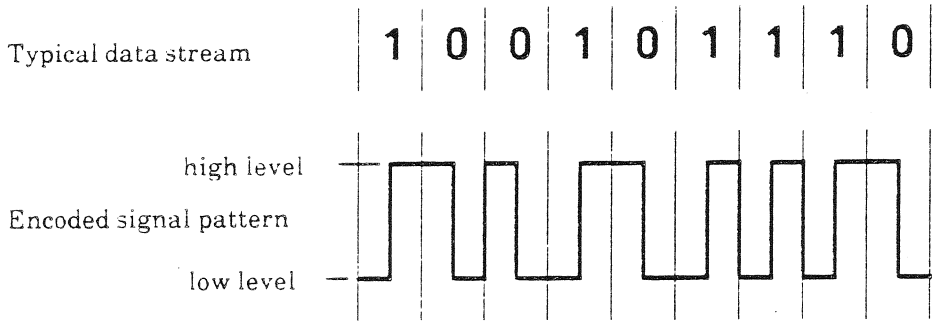


Figure 7-6: Manchester Encoding

7.5.1.2 Decoder

The decoder is used to separate the incoming phase encoded bit stream into a data stream and a clock signal. The decoder must be able to provide data and clock signals usable by the data link under the timing distortion imposed by the worst case system configuration. The decoder must provide usable output (clock and data) after no more than 16 bit cell times following reception of an encoded signal (the assertion of *carrier sense*). The first signals received from the transceiver at the beginning of frame reception may not constitute a valid, properly encoded bit; it is possible for the time from the first transition seen to the first true mid-bit cell transition to assume any value from zero to 100 ns.

The decoder input is normally derived from the coaxial cable, through the transceiver cable receive pair. It is not necessary for the decoder to provide usable output when there is a collision on the coaxial cable, regardless of whether the station using that decoder is involved in the collision.

7.5.1.3 Preamble Generation

Because most of the channel circuitry is allowed to provide valid output some number of bit times after being presented valid input, it is necessary for a preamble to be sent before the start of data link information, to allow the channel circuitry to reach its steady-state, with valid outputs throughout the system. Upon request by the data link to transmit the first bit of a new frame, the channel shall first transmit the preamble, a predetermined bit sequence used for channel stabilization and synchronization. If, while transmitting the preamble, the channel logic asserts the *collision detect* signal as specified in 7.5.2, any remaining preamble bits must be sent before proceeding with the transmission of the bit submitted by the data link.

The preamble is a 64-bit pattern to be presented to the channel encoder in the same manner as data link information. The pattern is:

10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011.

The bits are transmitted in order, from left to right. The nature of the pattern is such that when encoded, it appears as a periodic waveform on the cable, with a 5 MHz frequency. Excepting the final two bits, the only transitions present in the waveform are in the center of the bit cells. This is depicted in Figure 7-7. The last two bits of the preamble contain transitions at both the bit cell centers and the edges, and are used to indicate the end of the preamble and the beginning of the data link encapsulation portion of the frame. The next bit transmitted is the bit originally submitted for transmission by the data link.

Preamble removal on reception is discussed in 7.5.4.1.

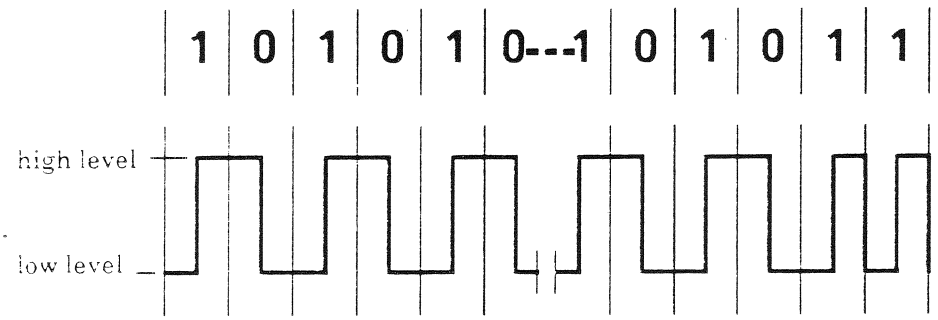


Figure 7-7: Preamble Encoding

7.5.2 Collision Detect Signal

While transmitting, the channel must indicate to the data link when the signals on the coaxial cable imply simultaneous transmission attempts by one or more other stations. This is normally indicated through the collision presence pair in the transceiver cable, described in 7.2.1.3.

The channel logic must assert the *collisionDetect* signal within 2 bit times (200 ns), following the onset of collision presence. A functional logic description of the *collisionDetect* signal is shown in Figure 7-8.

Following the loss of collision presence information, the channel must clear the *collisionDetect* signal within 1.6 bit cell times (160 ns).

Except for the collision presence test function specified in 7.4.7, the *collisionDetect* signal is undefined during the carrier sense inhibit period specified in 7.5.3. It may assume any value during this time. At other times the collision detect signal shall not be asserted unless there is a true collision on the channel.

7.5.3 Carrier Sense Signal

The channel must indicate to the data link the presence of carrier, a transmission

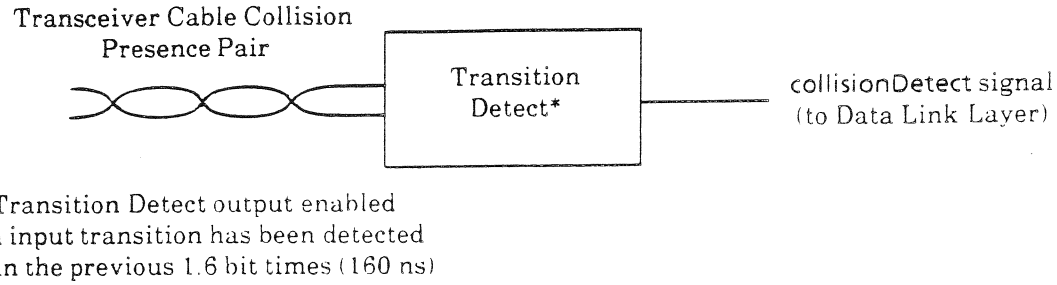


Figure 7-8: Functional Logic of collisionDetect Signal

attempt on the coaxial cable by a station. This is normally indicated through both the receive and collision presence pairs in the transceiver cable, described in 7.2.1.

The *carrierSense* signal shall be asserted when one or more stations are attempting transmission on the cable, regardless of whether the station sensing carrier is transmitting at that time. The channel logic must assert the *carrierSense* signal within 2 bit times (200 ns) following the onset of carrier presence information. A functional logic description of these signals is shown in Figure 7-9.

Following the loss of carrier presence information (receive transitions and collision presence information) the channel must clear the *carrierSense* signal within 1.6 bit cell times (160 ns). The *carrierSense* signal must not be asserted again for a period of 4 ms minimum (9.6 μs maximum), regardless of the state of the receive pair or collision presence pair during this time. Following this "inhibit period," normal operation of the carrier sense function must resume.

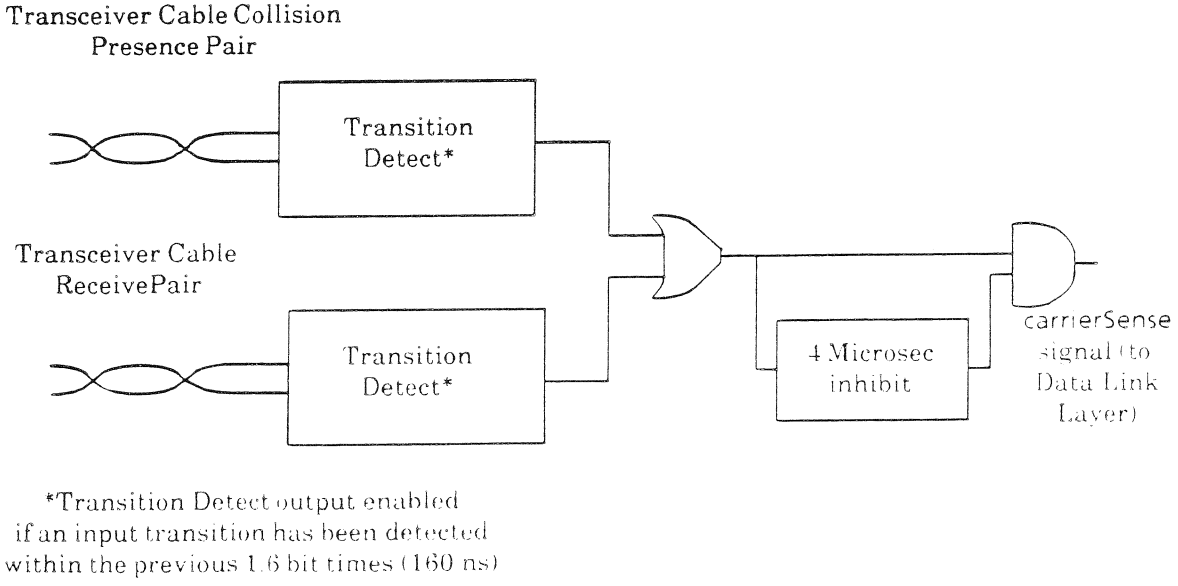


Figure 7-9: Functional Logic of carrierSense Signal

7.5.4 Channel Framing

During reception, the channel must provide the data link with signals to indicate beginning and end of frame.

7.5.4.1 Beginning-of-Frame Sequence

The channel logic 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. However, dependent upon the physical configuration of the system, there are some number of preamble bits to be received by the channel before the start of the data link frame as indicated by the two ones at the end of preamble (see Fig. 7-7). In addition, the first signals received from the decoder may be invalid due to the first bit allowance of the transceiver (see 7.4.2.2). The channel must wait until the decoder is providing valid clock and data (see 7.5.1.2) before monitoring the output of the decoder for the double-1 indicating end of preamble, and beginning of data link frame. Upon reception of the double-1, the channel shall begin passing successive bits to the data link through the defined receive bit stream interface. If, while properly monitoring for the double-1, a 'double-0' is encountered, the physical channel shall not pass any bits of the current frame to the data link. Normal operation of the data link and channel shall resume on the subsequent frame.

7.5.4.2 End-of-Frame Sequence

As specified in 7.5.3, the *carrier sense* signal must be cleared no later than 1.6 bit times (160 ns) after the cessation of activity on the coaxial cable as seen by the channel logic. The channel ensures that no extraneous bits will appear at the end of a frame following the last valid bit.

7.6 Channel Configuration Requirements

7.6.1 Cable Sectioning

The 500 meter maximum length coaxial cable segment need not be made from a single, homogeneous length of cable. The boundary between two cable sections (joined by coaxial connectors; two male plugs and a barrel) represents a signal reflection point due to the impedance discontinuity caused by the batch-to-batch impedance tolerance of the cable. Since the worst-case variation from 50 Ω is 2 Ω (see 7.3.1.1.1), a possible worst-case reflection of 4% may result from the joining of two cable sections. The configuration of long cable segments (up to 500 meters) from smaller sections must be made with care. The following recommendations apply, and are given in order of preference:

1. If possible, the total segment should be made from one homogeneous (no breaks) cable. This is feasible for short segments, and results in minimal reflections

from cable impedance discontinuities.

2. If cable segments must be built up from smaller sections, it is highly desirable to ensure that all the sections are from the same manufacturer and lot. This is equivalent to using a single cable, since the cable discontinuities are due to extruder limitations, and not extruder-to-extruder tolerances. There are no restrictions in cable sectioning if this method is used. However, if a cable section in such a system is later replaced, it must be replaced either with another cable from the same manufacturer and lot, or with one of the standard lengths described below.

3. If uncontrolled cable sections must be used in building up a longer segment, the lengths should be chosen such that reflections, when they occur, do not have a high probability of adding in phase. This can be accomplished by using lengths which are odd integral multiples of a half-wavelength in the cable at 5 MHz; this corresponds to using lengths of 23.4, 70.2, and 117 meters ($\pm$ 0.5 meters) for all sections. These are considered to be the standard lengths for all cable sections. Using these lengths exclusively, any mix or match of cable sections may be used to build up a 500 meter segment without incurring excessive reflections.

4. As a last resort, an arbitrary configuration of cable sections may be employed, if it has been confirmed by analysis or measurement that the worst-case signal reflection due to the impedance discontinuities at any point on the cable does not exceed 7% of the incident wave when driven by a transceiver meeting the specifications of 7.4.

7.6.2 Transceiver Placement

Transceivers and their associated connections to the cable cause signal reflections due to their non-infinite bridging impedance. While this impedance must be implemented as specified in 7.3.1.4 and 7.4.1, the placement of transceivers along the coaxial cable must also be controlled to insure that reflections from transceiver do not add in phase to a significant degree.

Coaxial cables marked as specified in 7.3.1.1.6 have marks at regular 2.5 meters spacing; a transceiver may be placed at any mark on the cable. This guarantees both a minimum spacing between transceivers of 2.5 meters, as well as controlling the relative spacing of transceivers to insure non-alignment on fractional wavelength boundaries.

The total number of transceivers on a cable segment shall not exceed 100.

7.6.3 System Earth Connection

The sheath conductor of the coaxial cable shall make electrical contact with earth reference at only one point. Redundant earth reference connections must be

avoided. Insulators may be used to cover any coaxial connectors used to join cable sections and terminators, to insure that this requirement is met. A sleeve or boot attached at installation time is acceptable.

The sheath conductor of the transceiver cable shall be connected to the earth reference or chassis of the device housing the station logic.

7.6.4 Repeaters

Repeaters are used to extend the channel length and topology beyond that which could be achieved by a single coaxial cable segment. (See the channel configuration model in 7.1.5.) A repeater requires a transceiver on each of the segments between which it is repeating signals. These transceivers must be as specified in 7.4, and must be counted towards the maximum specified in 7.6.2. These transceivers may be connected to the repeater by transceiver cables as specified in 7.2.

The specifications of this section apply to the repeater exclusive of transceivers and transceiver cables.

A repeater may incorporate a point-to-point link internal to the repeater together with appropriate electronics at each end of the link. See 7.1.5.

A maximum of two repeaters (or four half-repeaters, i.e. two remote repeaters) may be in the signal path between any two transceivers on the channel.

7.6.4.1 Carrier Detect and Transmit Repeat

A repeater implements the carrier sense function as specified in 7.5.3 for both segments between which it is connected. Upon detection of carrier from one segment (which becomes the incoming segment), the repeater must retransmit all received signals from that segment onto the other segment (which becomes the outgoing segment). Signals shall be retimed and amplified as specified in 7.6.4.3.

The repeater must begin presenting preamble bit transitions to the encoder driving the outgoing segment transceiver cable within 600 ns of the assertion of the carrierSense to the incoming segment's data link, not including delays associated with any internal point-to-point link present in the repeater. The repeater must transmit 64 preamble bits onto the outgoing segment.

7.6.4.2 Collision Detect and Collision Repeat

A repeater implements the collision detect function as specified in 7.5.2 for both segments between which it is connected. If, during the operation of the repeater, collision is detected on either segment between which it is connected, the repeater must ensure that all stations involved in the collision recognize the event as a collision, regardless of which segment the station is on. Upon detecting collision,

the repeater must begin presenting transitions as specified in 7.5.1 to the encoders driving the transceiver cables of both segments between which it is connected within 400 ns of the assertion of collisionDetect, not including delays associated with any internal point-to-point link present in the repeater. The repeater shall cease presenting transitions upon the deassertion of collisionDetect in the repeater.

7.6.4.3 Repeater Signal Regeneration

7.6.4.3.1 Signal Amplification

The repeater (with its associated transceivers) shall ensure that any signals repeated between segments shall have the same amplitude characteristics at the transceiver output of the repeated-to segment as they did at the output of the transmitter on the repeated-from segment, allowing for transceiver output tolerances as specified in 7.4.1.3. Any loss of signal-to-noise ratio due to cable loss and noise pickup is thus regained at the output of the repeater.

7.6.4.3.2 Signal Timing

The repeater must ensure that the timing characteristics of the signals at the transceiver output of the repeated-to segment are the same as those at the output of the transmitter on the repeated-from segment, allowing for transceiver and transceiver cable tolerances. Any timing distortion due to transceivers and cable distortion is thus regained at the output of the repeater.

7.7 Environment Specifications

The following sections specify the physical environment in which all channel components must operate to be considered compatible.

7.7.1 Electromagnetic Environment

The physical channel hardware shall meet its specifications when operating in the following ambient plane-wave fields:

2 Volts/Meter from 10 KHz through 30 MHz

5 Volts/Meter from 30 MHz through 1 GHz

7.7.2 Temperature and Humidity

All physical channel hardware, with the possible exception of the channel logic components, shall operate over the ambient temperature range of 5 to 50 degrees Celsius, and relative humidity range of 10% to 95% non-condensing. The channel logic components are normally part of the station hardware, and are thus subject to individual station product requirements. Hardware which does not meet the temperature and humidity requirements specified must state so in its published product specification.

8. ETHERNET CONFIGURATION TESTING PROTOCOL

The *Ethernet Configuration Testing Protocol* provides a minimum testing capability of communication between stations on an Ethernet. It is the only Client Layer protocol specified in this document and has the only assigned Ethernet type field value in this document. All Ethernet stations must support the configuration testing functions.

8.1 Goals

The goals of the *Ethernet Configuration Testing Protocol* specification are:

- Provide the required functions to enable all forms of multi-station loop tests that are necessary to diagnose a station's ability to communicate.
- Allow each station to assume the responsibility to diagnose its own ability to communicate.
- Allow a network management station to diagnose any other station's ability to communicate.
- Minimize processing and memory requirements.

8.1.1 Configuration Testing Functions

A station using the *Configuration Testing Protocol* can ascertain the following:

- The ability to communicate with a specific remote station.
- The ability to communicate with some remote station.
- With the help of a third party station, the ability to hear or to be heard by a specific station.

8.1.2 Conformance Requirements

In order to support the above functions and to provide for communication checking by a network management entity, all Ethernet stations must implement a *Configuration Server* module with the following capabilities.

The *Configuration Server* receives datagrams addressed to its station's Ethernet physical address and to the Ethernet broadcast address whenever the Data Link is turned on.

The *Configuration Server* must receive and process datagrams as large as the largest datagram transmitted or received by any client layer module at that station. It must also receive and process datagrams having an Ethernet Data field

(i.e. loopback protocol frame) up to and including 512 octets long.

As an option, the *Configuration Server* may also receive these datagrams on a fixed multicast address called the *loopback assistance multicast address*, in which case the *Configuration Server* is called a *loopback assistant*. Those systems implementing this option are said to belong to the *loopback assistance group*.

This loopback assist multicast address has a fixed value of CF-00-00-00-00-00.

8.2 Functions, Loopback Frames and Protocol Messages

Loopback protocol frames hop between two or more stations until they reach their final destination. The station originating a loopback frame defines its whole route. A loopback frame contains some *loopback protocol messages* encapsulated within each other, as described in Sections 8.3 and 8.4.

There are the following two types of messages in this protocol.

1. Reply

   A message homebound to its final destination, i.e. the originating station; therefore this message requires no response.

2. Forward Data

   A message to be forwarded to another station.

Multiple hop frames consist of some *Forward Data* messages and one *Reply* message. The *Reply* message is always the last. In practice, the final destination of a loopback frame is always the originating station and the minimum path consists of two hops: from the sender to the receiver and back to the sender.

In the interest of simplicity and efficiency, loopback operation and message formats are designed to meet the following requirements:

1. All fields begin on 16-bit boundaries.

2. Progressive operations on the same frame (e.g., looping it back) do not change the size.

This second objective is met by encapsulating various loopback messages within each other. In addition to that encapsulation, loopback protocol frames must be encapsulated into an Ethernet packet for transmission on the Ethernet. It is important to notice that we are dealing with two different concepts of encapsulation: one is between messages belonging to the same protocol; the other is the encapsulation of message packets of a *higher* level into packets of a *lower* level in the hierarchy of protocols (see [4]).

8.3 Encapsulation of Loopback Protocol Frames for Transmission on an Ethernet

The fact that this protocol is a *higher level*, Client Layer protocol, means that any frames of this protocol will be passed down to the Data Link which will encapsulate them in an Ethernet frame. On reception, the Data Link will decapsulate them. Figure 8-1 shows how a loopback protocol packet (see Section 8.4 for details) is encapsulated in an Ethernet packet (like the one shown in figure 6-1).

In the type field of the Ethernet packet is the protocol type number associated with the *Configuration Test Protocol*, i.e., 90-00. See Section 6.2 for details.

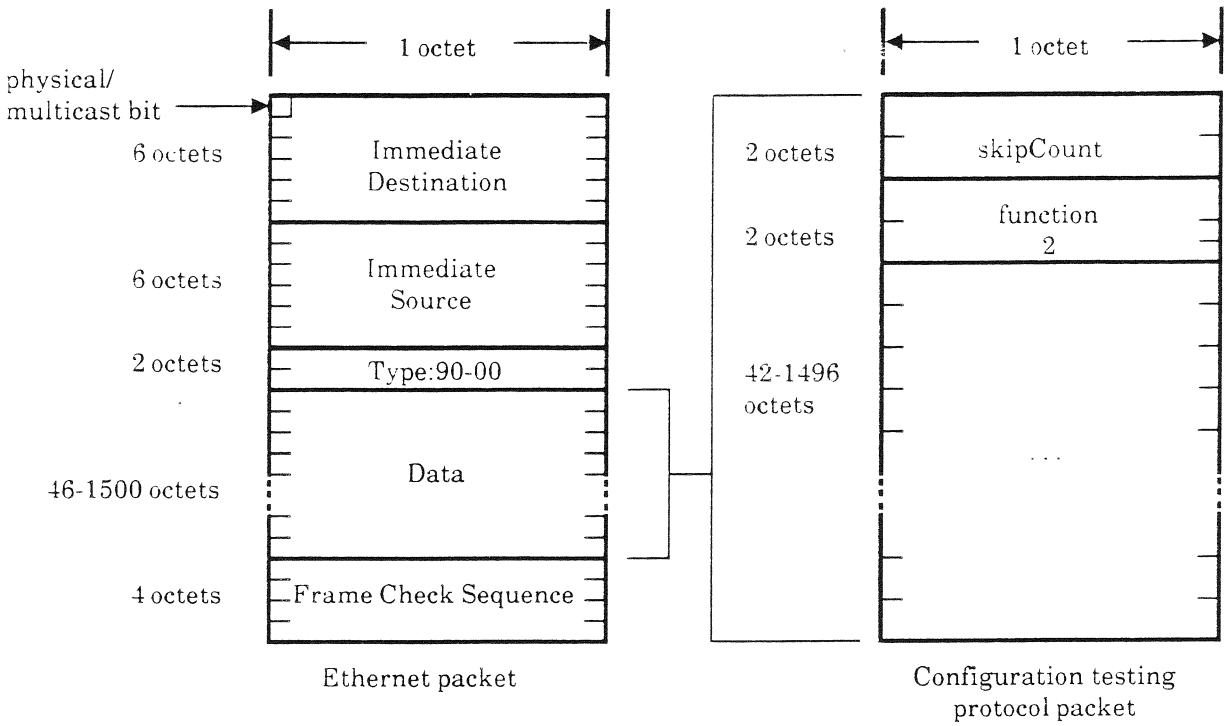


Figure 8-1: A Loopback Frame Encapsulated in an Ethernet Packet

8.4 Frame and Message Formats

This section defines the format of the *Configuration Testing Protocol* frames and messages. The descriptive and transmission conventions are the same as for that in Sections 5, 6, and 7. The usage of these formats is described in Ssection 8.5. The skipCount and function code fields are 16 bit binary integers; the order of transmission is: least significant octet first, i.e. from the *least significant bit* to the *most significant bit* of the *least significant octet.*

A *Configuration Testing Protocol Frame* begins with a two octet field called the

skipCount. After this field there are some *Configuration Testing Protocol Messages* encapsulated within each other; the message begins with a two octet function code. The values of the function codes are the following:

Function code	Function
1	Reply
2	Forward Data

8.4.1 Reply Message

This message is recognized as a loopback reply. The message has the following format: 1) the first two octets show the function code, i.e. the value 1; 2) the next two octets are used for a receipt number; and 3) all the remaining octets contain any arbitrary data. See the right hand side packet in figure 8-2.

8.4.2 Forward Data Message

This message gets forwarded to some other station. The message has the following format: 1) two octets for the function code, i.e. the value 2; 2) six octets for the forwarding address; 3) the data.

The forwarding address must be physical, and the data must consist of another message. Figure 8-2 shows a loopback protocol frame consisting of a Reply message encapsulated in a Forward Data message.

8.4.2.1 Restrictions on Forward Data Messages

If a Forward Data message is directed to the broadcast address or to any multicast address, this message should not encapsulate another Forward Data message. The *Configuration Server* may, optionally, discard messages that violate this restriction.

8.5 Operation of the Protocol

During the entire life time of a *Configuration Test Protocol* frame, the only field that changes is the skipCount, i.e. the first two octets. Its value indicates how many octets to skip after the skipCount to find the function code of the message to be processed. When a Forward Data message is processed, the processing station updates the skipCount so that the next recipient will process the next following encapsulated message. Note that in order to meet the 16-bit boundary requirement, the skipCount is always an even number. Notice also that only the first two octets of the frame are altered during successive forwarding operations, while the remainder of the frame remains constant.

The operational descriptions assume the following Configuration Test Protocol

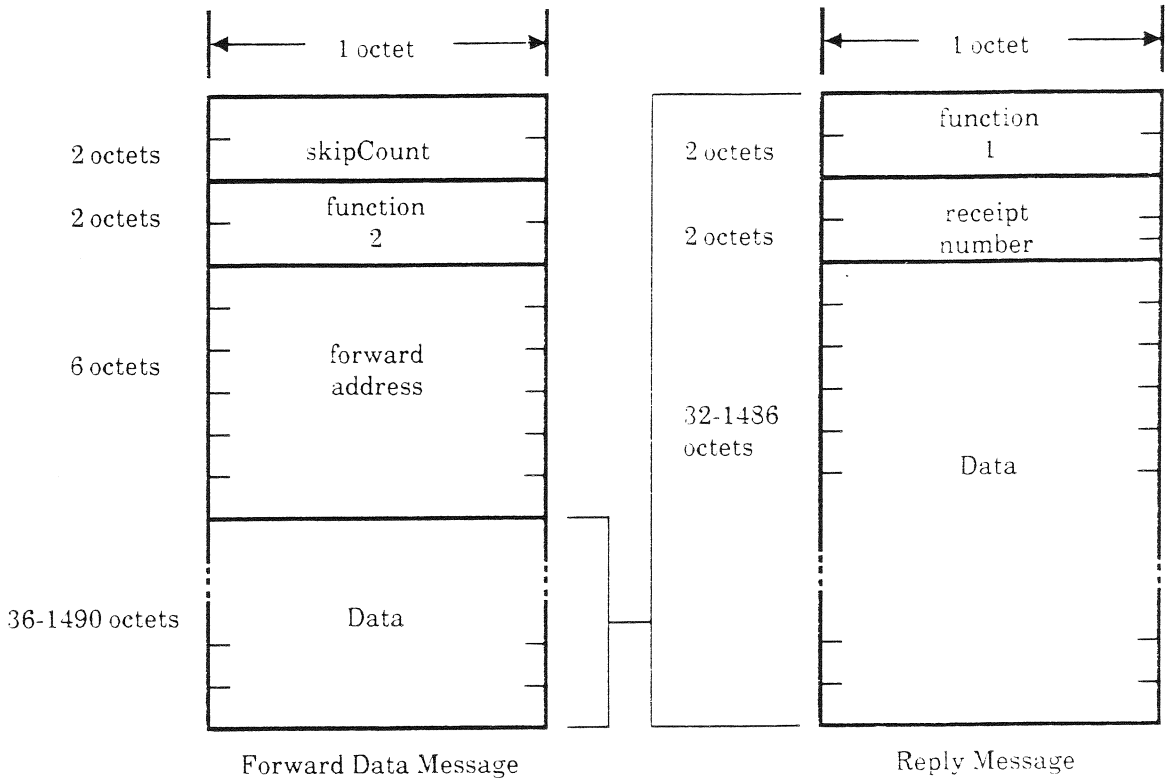


Figure 8-2: A Reply Message Encapsulated in a Forward Data Message

message types:

1. Reply

   A message homebound to its final destination, i.e., the originating station; therefore this message requires no response.

2. Forward Data

   A message to be forwarded to another station.

The *Configuration Server* must respond to the data link within one second of the time it receives a valid message.

The received frame is processed according to the message function code:

1. Reply message

   The message is delivered to a higher level process.

2. Forward Data message

The value of the skip count is increased by the length of the function code and forwarding address (8 octets). If the forward address is a multicast address or the broadcast address, the message is ignored. Otherwise, the message is transmitted to the forward address.

8.6 Usage Examples

The following examples address the application of the *Configuration Testing Protocol* functions, and are intended as examples of how a higher level process can use the facilities. They are neither a specification for how they must be used, nor an exhaustive test script.

8.6.1 Local Control Test Example

In this case, a station is unable to communicate with another station it believes is on the network and operational. The following test script can be used by the testing station to investigate the problem.

First, the testing station attempts to communicate directly with the suspect station using the simplest, two hop frame consisting of: a) a Forward Data message addressed to the suspect station, and b) a Reply message, encapsulated in the former, addressed to the testing station. If the frame is returned, communication is possible and the problem may be either intermittent; for example, the suspect station may have been down or busy, or there may be a problem with message length or data pattern. Different message lengths and/or data patterns could then be tried. On the other hand, if communication fails the test could be repeated a few times to reduce the probability of a transient error.

If the above test fails, use some other station as an assistant and attempt the same series of operations. The datagram is constructed as follows: a) a Forward Data message to the assistant station, containing: b) a Forward Data message to the suspect station, containing: c) a Forward Data message to the assistant station, containing: d) a Reply message to the testing station. This four-message datagram obtains the full assistance of the loopback assistant.

If no assistant address is known, send a simple loopback request (i.e. a two hop frame) to the loop assistant multicast address and use the source address of any responding station as an assistant address. If there is no response to the loop assistant multicast address, the broadcast address can be tried as a last resort.

If an assistant station can communicate with the suspect station but the testing station cannot, the testing station can test for the direction in which communication does not work. The assistant station can be used to detect either transmit or receive problems. By repeating the above test with different suspect stations, it can be determined whether the testing station or the suspect station is at fault, and isolate the problem to a particular transmitter or receiver.

In order to determine whether the suspect station can receive datagrams sent by the testing station, use the following datagram: a) a Forward Data message to the suspect station, containing: b) a Forward Data message to the assistant station, containing: c) a Reply message to the testing station.

Similarly, use the following datagram to test whether the suspect station can successfully transmit to the testing station: a) a Forward Data message to the assistant station, containing: b) a Forward Data message to the suspect station, containing: c) a Reply message to the testing station.

When a station finds itself unable to communicate, it can report this in whatever way is available to it. An operator or control center can then respond by attempting to isolate and repair the problem.

8.6.2 Remote Control Test Example

When a control center receives a report that a station cannot communicate properly, it can investigate the problem. It can first diagnose its own ability to communicate with the reporting station. If this communication appears to work, the control center can similarly check its ability to communicate with the remote station that the reporting station could not reach.

If the control center can communicate with both stations, it can then use the full assistance frame, using either station as the assistant, in order to see if they can communicate with each other. If they cannot, the control center can attempt to isolate the problem using transmit and receive assistance.

APPENDIX A: GLOSSARY

This section defines some of the essential terminology associated with the Ethernet.

baseband coaxial system: A system whereby information is directly encoded and impressed on the coaxial transmission medium. One information signal at a time can be present on the medium without disruption (see collision).

binary exponential backoff: The algorithm used to schedule retransmissions after a collision. So called because the interval from which the retransmission time is selected is expanded exponentially with repeated collisions.

broadcast: Describes the class of media for which the Ethernet is designed, in which all stations are capable of receiving a signal transmitted by any other station. Also, describes the mode of usage of such a medium by the Data Link Layer in which all stations are instructed to receive a given frame.

carrier sense: A signal provided by the Physical Layer to the Data Link Layer to indicate that one or more stations are currently transmitting on the channel.

channel logic: The logical functions provided between the transceiver cable and the Data Link, which support the defined interface between the data link and the physical layers.

Client Layer: Collective term used to describe any layer of a network architecture, which use the Ethernet Data Link and Client interface.

coaxial cable: A two-conductor, concentric, constant impedance transmission line.

coaxial cable interface: The electrical, mechanical, and logical interface to the shared coaxial cable medium. This is a mandatory compatibility interface, which must be correctly implemented by every Ethernet implementation.

coaxial cable section: An unbroken piece of coaxial cable, fitted with coaxial connectors at its ends, used to build up coaxial cable segments.

coaxial cable segment: A length of coaxial cable made up from one or more coaxial cable sections and coaxial connectors, terminated at each end in its characteristic impedance. A 500 meter segment is the longest configuration possible without repeaters.

collision: The result of multiple transmissions overlapping in the physical channel, resulting in garbled data and necessitating retransmission.

collision detect: A signal provided by the Physical Layer to the Data Link Layer to indicate that one or more other stations are contending with the local station's

transmission. It can be true only during transmission.

collision enforcement: Transmission of extra, encoded "jam" bits after a collision is detected, to insure that the duration of the collision is sufficient to guarantee its detection by all transmitting stations.

compatibility interfaces: The coaxial cable interface, and the transceiver cable interface, the two points at which hardware compatibility is defined to allow connection of independently designed and manufactured components to the Ethernet.

contention: Interference between colliding transmissions (see collision). Resolution of occasional contention is a normal part of the Ethernet's distributed link management procedure (see CSMA-CD).

controller: The implementation unit which connects a station to the Ethernet, typically comprising part of the Physical Layer, much or all of the Data Link Layer, and appropriate electronics for interfacing to the station.

CSMA-CD: Carrier Sense Multiple Access with Collision Detection, the generic term for the class of link management procedure used by the Ethernet. So called because it a) allows multiple stations to access the broadcast channel at will, b) avoids contention via carrier sense and deference, and c) resolves contention via collision detection and retransmission.

Data Link Layer: The higher of the two layers in the Ethernet design, which implements a medium-independent link level communication facility on top of the physical channel provided by the Physical Layer.

deference: A process by which a data link controller delays its transmission when the channel is busy to avoid contention with ongoing transmissions.

frame check sequence: An encoded value appended to each frame by the Data Link Layer to allow detection of transmission errors in the physical channel.

interframe spacing: An enforced idle time between transmission of successive frames to allow receiving data link controllers and the physical channel to recover.

jam: An encoded bit sequence used for collision enforcement.

Manchester encoding: A means by which separate data and clock signals can be combined into a single, self-synchronizable data stream, suitable for transmission on a serial channel.

multicast: An addressing mode in which a given frame is targeted to a group of logically related stations.

physical address: The unique address value associated with a given station on the

network. An Ethernet physical address is defined to be distinct from all other physical addresses on all Ethernets.

Physical Channel: The implementation of the physical layer.

Physical Layer: The lower of the two layers of the Ethernet design, implemented by the physical channel using the specified coaxial cable medium. The Physical Layer insulates the Data Link Layer from medium-dependent physical characteristics.

preamble: A sequence of 64 encoded bits which the Physical Layer transmits before each frame to allow synchronization of clocks and other Physical Layer circuitry at other sites on the channel.

repeater: A device used to extend the length and topology of the physical channel beyond that imposed by a single segment, up to the maximum allowable end-to-end channel length.

round-trip propagation time: In bit times, the time required in the worst-case for a transmitting station's collision detect signal to be asserted due to normal contention for the channel. This delay is the primary component of the slot time.

slot time: A multi-purpose parameter which describes the contention behavior of the Data Link Layer. It serves as (a) an upper bound on the collision vulnerability of a given transmission, (b) an upper bound on the size of the frame fragment produced by a collision, and (c) the scheduling quantum for collision retransmission.

station: A single addressable site on the Ethernet, generally implemented as a computer and appropriate peripherals, and connected to the Ethernet via a controller and a transceiver.

transceiver: The portion of the Physical Layer implementation that connects directly to the coaxial cable and provides both the electronics which send and receive the encoded signals on the cable and the required electrical isolation.

transceiver cable: A four pair, shielded cable used for the transceiver cable interface.

transceiver cable interface: The electrical, mechanical and logical interface which connects the transceiver to the controller. The standard transceiver cable is a recommended compatibility interface.

APPENDIX B: NOTES ON ADDRESS AND TYPE ASSIGNMENT, AND LICENSING

Address and Type Assignment

The address and type fields will be administered by Xerox Corporation.

A block of addresses will be assigned to each licensee of Ethernet patents (see below). Others may obtain an address block or type field assignment by request. A nominal fee to cover administrative costs will be charged.

Submit written requests to:

Xerox Corporation
Ethernet Address Administration Office
3333 Coyote Hill Road
Palo Alto, CA 94304

As specified in Section 6.2, Ethernet addresses are unique sequences of 48 bits. An address may be represented as a string of six octets, A-B-C-D-E-F. The least significant bit of the first octet, A, is the multicast bit (therefore A in physical Ethernet addresses is an even number). The numbers A-B-C are assigned by the Ethernet Address Administration Office at the above address. The remaining octets, D-E-F are assigned locally by the organization assigned the blocks identified by A-B-C.

NOTE: As defined in Section 6.2, bits are transmitted on the coaxial cable with the low-order (20-th) bit for each octet transmitted first.

The Ethernet Address Administration Office has instituted procedures which ensure that the block assignments are unique. It is strongly recommended that companies making Ethernet-compatible equipment utilize procedures that ensure the unique assignment of addresses.

NOTE: In no way should these addresses be used as product codes; i.e, for the purpose of aiding company inventory procedures.

Ethernet Address Checksum

When Ethernet addresses must be manually transcribed, it is desirable to append a checksum to detect transcription errors. The use of a checksum is optional. However, if a checksum is used for transcription purposes, it must conform to the following specification. This checksum may also be used inside stations to verify the integrity of stored Ethernet addresses.

Let an Ethernet address be given, using the standard notation, by a string of octets A-B-C-D-E-F where each octet is a number from 0 to 225. Then the checksum is a 16-bit value, K, calculated as follows:

$$K = [A \cdot 2^{10} + B\,2^2 + C \cdot 2^9 + D \cdot 2^1 + E \cdot 2^8 + F] \bmod 2^{16} - 1.$$

When inputting or displaying an address together with its checksum, the address is in the form A-B-C-D-E-F-G-H, where $K = G \cdot 2^8 + H$.

The checksum can be calculated using 16-bit one's complement binary arithmetic, shifting left prior to adding each pair of octets, as illustrated by the following Pascal program:

```
var A, B, C, D, E, F, G, H, I, K;
var Octet: array [0..5] of integer;
Octet[0]: = A; Octet[1]: = B; Octet[2]: = C;
Octet[3]: = D; Octet [4]: = E; Octet [5]: = F;
K : = 0;
for I = 0 to 2 do
   begin
      K: = 2*K;
      if K > = 65536 then K : = K - 65535;
      K: = K + 256 *Octet[2*I] + Octet[2*I + 1];
      if K > = 65536 then K : = K - 65535;
   end;
if K = 65535 then K : = 0;
G : = K div 256;
H : = K mod 256
```

For example, the Ethernet address FO-2E-15-6C-77-9B would be written with checksum as FO-2E-15-6C-77-9B-63-2F.

Ethernet type fields are pairs of octets, and are also under the control of the Ethernet Address Administration Office. A catalog of assigned addresses and type field assignments is maintained by the Ethernet Address Administration Office. Public type field assignments are published periodically and are maintained under separate administrative procedures.

Licensing

Ethernet incorporates features that are protected by one or more patents assigned to Xerox Corporation. Questions on the need for licensing particular uses of this specification should be directed to:

Xerox Corporation
Director of Licensing
Stamford, CT 06904

APPENDIX C: CRC IMPLEMENTATION

Every frame contains, in its frame check sequence field, a 32-bit cyclic redundancy check (CRC) code. Because the formal mathematical definition of this code (see 6.2.4) is not suggestive of an appropriate implementation, this appendix outlines one possible implementation in terms of a feedback shift register. This type of implementation is likely to be common in practice, but is not a mandatory part of the specification.

The feedback shift register (see Figure C-1) is used to represent division of the pre-scaled message by the generating polynomial. The 32-bit register is accessed via the three signals Input, Output, and Control. When Control = 1, Input bits are shifted into the feedback shift register and also fed directly back to Output. When Control = 0, the feedback paths are disabled and the shift register shifts the complement of its contents to Output.

Before CRC generation at the transmitting end, initialization logic (not shown in Figure C-1) preloads the shift register to all 1's. Control is then held at 1 while the address, type and data fields of the outgoing frame are shifted into Input and the CRC is generated. Meanwhile, the same bits emerging at Output are transmitted over the network. When the last bit of the data field has been processed, Control is set to 0 and the complemented CRC is shifted out for transmission, starting with the x^{31} term (see 6.2.4).

CRC checking at the receiving end also begins with the shift register preloaded to all 1's. Control is then held at 1 while the incoming bits are shifted into Input to regenerate the CRC. When the last bit of the data field has been processed, the shift register should contain the CRC whose binary complement is about to arrive on the network. Since this field boundary cannot be recognized by the receiver, however, Control remains at 1 and the bits of the CRC continue to feed into the the shift register until the end of the entire frame is reached. If the two CRCs match, the final contents of the shift register is the value:

11000111 00000100 1101110l 01111011

(where the leftmost bit corresponds to the x^{31} term of the polynomial and the rightmost to the x^0 term). Any other final value indicates a detected error. (The extra logic to test for this value is not shown in Figure C-1.)

One potential problem which is avoided in this implementation is insensitivity of the shift register to incoming zero-bits when it is in the all-zero state. Following standard practice, this state is avoided at the beginning and end of the frame by preloading the shift register with all 1-bits, and by inverting each bit of the final CRC. Logically, these correspond, respectively, to the complementing of the first 32 bits of the frame and to the final complementing of the remainder, as specified in the mathematical definition in 6.2.4. See also [9] for further discussion.

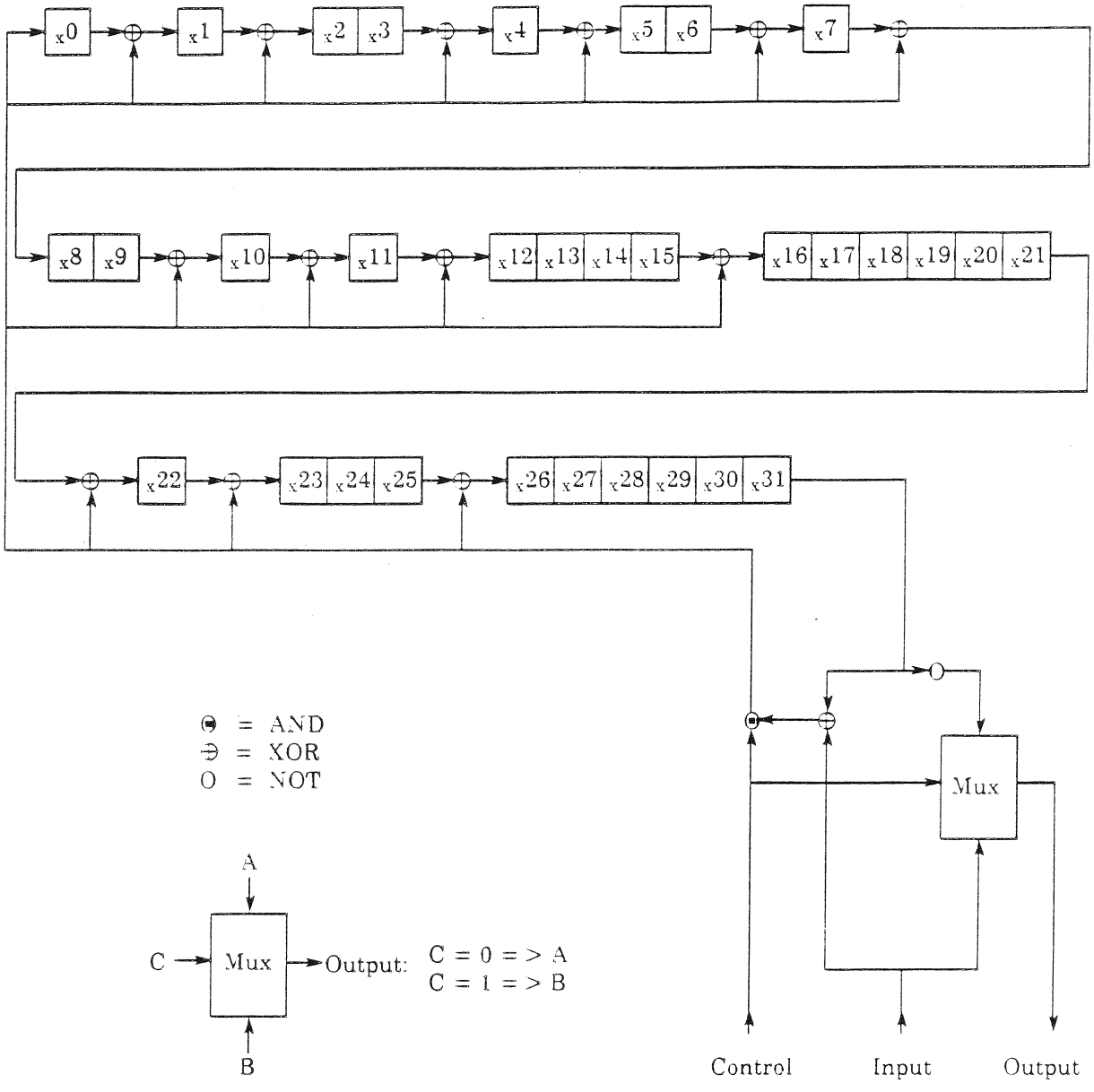


Figure C-1: CRC Implementation

APPENDIX D: IMPLEMENTATION OF TRANSCEIVER CABLE DRIVER AND RECEIVER

This appendix presents circuit diagrams for typical implementations of the transceiver cable drivers and receivers. The use of these exact circuits is not necessary for conformance to the specification; equivalent circuits may be used as long as the relevant specifications are met.

Figure D-1 depicts an implementation of the transceiver cable driver specified in 7.2.4. It is suitable for use at either end of the transceiver cable, as necessary; i.e., it would be located at the station end to drive the transmit pair, and at the transceiver end to drive the receive and collision presence pairs. In addition, it is capable of driving suitable isolation circuits required to be located within the transceiver.

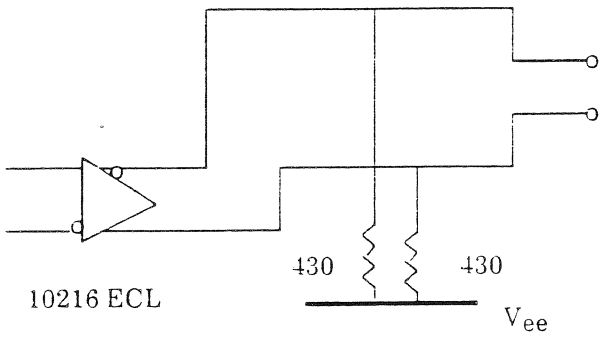


Figure D-1: Typical Transceiver Cable Driver

Figure D-2 depicts an implementation of the transceiver cable receiver specified in 7.2.5. It is suitable for use at either end of the transceiver cable, as necessary; i.e., it would be located at the station end to receive from the receive and collision presence pairs, and at the transceiver to receive from the transmit pair. It is capable of operating through suitable isolation circuits required to be located within the transceiver.

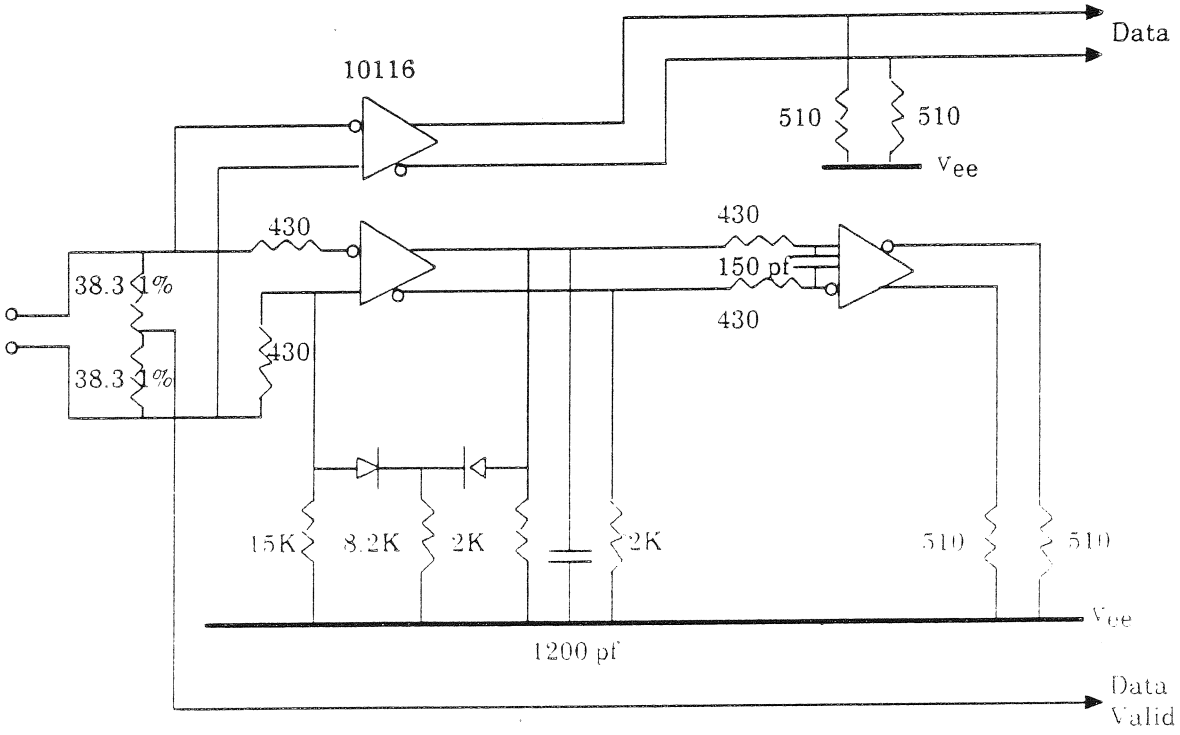


Figure D-2: Typical Transceiver Cable Receiver

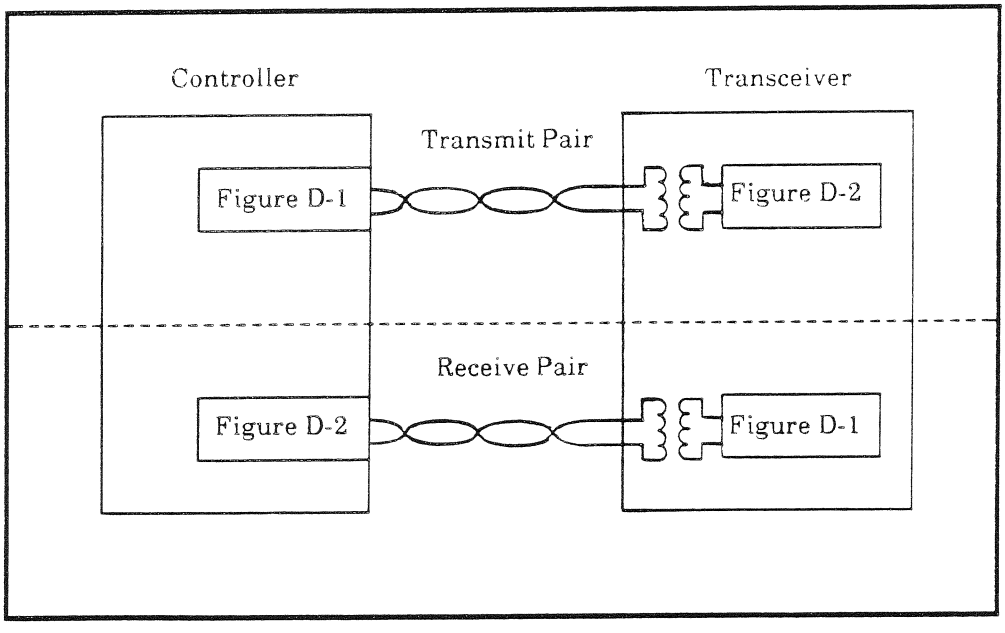


Figure D-3: Shows Relationship Between Figure D-1 and Figure D-2

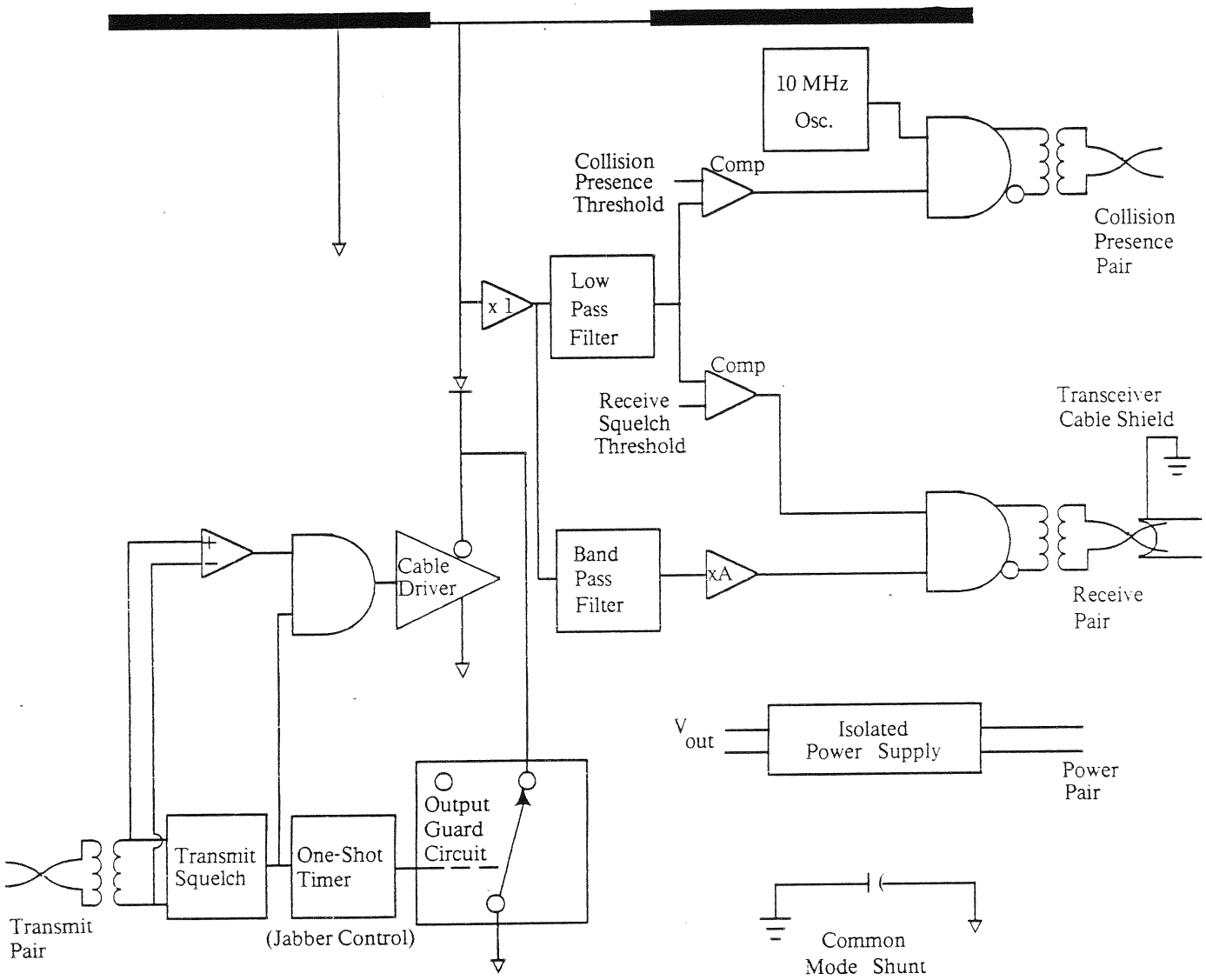


Figure D-4: Typical Transceiver Block Diagram

APPENDIX E: INTERFRAME RECOVERY

It is important that the data link controller implementations be able to receive a frame that arrives immediately after another frame has been transmitted or received. Here, "immediately" means 9.6 μsec, based on the minimum interframe spacing provided as recovery time for the data link. (See 6.3.2.2) It is important that the data link controller be able to resume reception within that time.

Reception of multiple closely spaced incoming frames is a very desirable capability, and is crucial for stations which tend to communicate with several other stations concurrently. There is one important case in which a data link controller implementation cannot reasonably be expected to receive closely spaced incoming frames: if the station hardware (e.g., I/O bus) is intrinsically unable to accept the bits of a frame at the rate at which they arrive over the network, each incoming frame must be buffered to allow the station to accept it at some lower rate. Assuming limited buffering resources (e.g., a one frame buffer), reception of subsequent frames cannot occur until sufficient buffer space is available. This mode of operation is allowed for low performance stations.

Reception of an incoming frame immediately after transmission of an outgoing frame is a very important capability, even for stations which do not tend to communicate with several other stations concurrently. All stations, low performance to high performance, should allow reception of an incoming frame immediately after transmission of an outgoing frame.

APPENDIX F: RECOGNITION OF MULTICAST GROUP ADDRESSES

The intent of multicast group addresses is that such addresses should correspond to groups of logically related Ethernet stations. Blocks of multicast group addresses are obtained in accordance with the procedures described in Appendix B. At any given time, one or more of these multicast addresses may be *active* on a given station, representing its membership in some number of multicast groups.

In order for this mechanism to be useful for a variety of applications, it is important that the recognition of active multicast group addresses be implemented very efficiently. In general, implementations of this filtering process will use some combination of hardware, software, and/or microcode. The fact that filtering of multicast addresses is architecturally considered to be a function of the Client layer should not be taken as detracting from the need for such efficiency.