Office Products Division

Evolution of the Ethernet Local Computer Network

by John F. Shoch, Yogen K. Dalal, Ronald C. Crane, and David D. Redell

XEROX

Evolution of the Ethernet Local Computer Network

by John F. Shoch, Yogen K. Dalal, Ronald C. Crane[1], and David D. Redell

OPD-T8102 September 1981

Abstract: The Ethernet system is a local computer network that supports high-speed packet switching in a locally distributed computing environment. This paper highlights some important elements of the basic system, and traces the evolution of the design from a research prototype to the specification of a multi-company standard. Starting from the Experimental Ethernet, we discuss tradeoffs among alternative implementations and design strategies which led to the Ethernet Specification. A concise summary of the Ethernet Specification is also included.

CR Categories: 3.81.

Key words and phrases: Local computer networks, Ethernet system, carrier sense multiple access with collision detection (CSMA/CD), communications standards.

This paper is to appear in the IEEE *Computer* Magazine.

[1] Present address: 3Com Corporation, 1390 Shorebird Way, Mountain View, California 94043.

XEROX
OFFICE PRODUCTS DIVISION and
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

1. Introduction

With the continuing decline in the cost of computing, we have witnessed a dramatic increase in the number of independent computer systems—systems used in such applications as scientific computing, business, process control, word processing, and personal computing. These machines do not compute in isolation, and with their proliferation comes a need for suitable communication networks—particularly *local computer networks* that can interconnect locally distributed computing systems. While there is no single definition of a local computer network, there is a broad set of requirements:

- relatively high data rates (typically 1 to 10 megabits per second);
- geographic distance spanning at most 1 kilometer (typically within a building or a small set of buildings);
- ability to support several hundred independent devices;
- simplicity, or the ability "to provide the simplest possible mechanisms that have the required functionality and performance"; [Crane and Taft, 1980]
- good error characteristics, good reliability, and minimal dependence upon any centralized components or control;
- efficient use of shared resources, particularly the communications network itself;
- stability under high load;
- fair access to the system by all devices;
- easy installation of a small system, with graceful growth as the system evolves;
- ease of re-configuration and maintenance;
- low cost.

One of the more successful designs for a system of this kind is the Ethernet local computer network [Metcalfe and Boggs, 1976; Metcalfe, *et al.*, 1977; Shoch and Hupp, 1979, 1980a; Crane and Taft, 1980]. Ethernet installations have been in use for many years. They support hundreds of stations, or hosts, and meet the requirements listed above.

In general terms, the Ethernet is a multi-access, packet-switched communications system for carrying digital data among locally distributed computing systems. The shared communications channel in an Ethernet is a passive broadcast medium with no central control; packet address recognition in each station is used to take packets from the channel. Access to the channel by stations wishing to transmit is coordinated in a distributed fashion, by the stations themselves, using a statistical arbitration scheme.

The Ethernet strategy can be used on many different broadcast media, but our major focus has been on the use of coaxial cable as the shared transmission medium. The *Experimental Ethernet* system was developed at the Xerox Palo Alto Research Center starting around 1972. Since then, numerous other organizations have developed and built "Ethernet-like" local networks [Shoch, 1980]. More recently, a cooperative effort involving Digital Equipment Corporation, Intel

Corporation, and Xerox Corporation has produced an updated version of the Ethernet design, generally known as the *Ethernet Specification* [Ethernet, 1980].

One of the primary goals of the Ethernet Specification is *compatibility*—providing enough information for different manufacturers to build widely differing machines in such a way that they can directly communicate with one another. It might be tempting to view the Specification as simply a "design handbook" that will allow designers to develop their own Ethernet-like network, perhaps customized for some specific requirements or local constraints. But this would miss the major point: successful interconnection of heterogeneous machines requires equipment that precisely matches *one* single specification.

Meeting the Specification is one necessary but not sufficient condition for inter-machine communication at all levels of the network architecture. There are many levels of protocol, such as transport, name binding, and file transfer, which must also be agreed upon and implemented in order to provide useful services [Boggs, *et al.*, 1980; Zimmermann, 1980; Dalal, 1981]. This is analogous to the telephone system: the common low-level specifications for telephony make it possible to dial from the US to France, but this is not of much use if the caller speaks only English while the person who answers the phone speaks nothing but French. Specification of these additional protocols is an important area for further work.

The design of any local network must be considered in the context of a distributed system architecture. Although the Ethernet Specification does not directly address issues of high-level network architecture, we view the local network as one component in an *internetwork* system, providing communications services to many diverse devices connected to different networks (see, for example, [Boggs, *et al.*, 1980; Cerf and Kirstein, 1978]). The services provided by the Ethernet are influenced by these broader architectural considerations, and we shall touch upon them briefly.

In the sections that follow, we highlight important design considerations, and trace the evolution of the Ethernet from research prototype to multi-company standard; we use the term Experimental Ethernet for the former, and Ethernet or Ethernet Specification for the latter. The term Ethernet is also used to describe design principles common to both systems. Section 2 provides an overview of Ethernet principles and components, and briefly describes both the Experimental Ethernet and the Ethernet Specification. Sections 3, 4, 5 and 6 discuss some of the specific design issues and tradeoffs between alternative implementations.

2. General Description of Ethernet-Class Systems

This section describes both the fundamental principles underlying the operation of the Ethernet, and the structure of such a system. In addition, we summarize the characteristics of two specific implementations of the Ethernet design: the Experimental Ethernet and the Ethernet Specification. These summaries should be sufficient for the discussion of design issues, tradeoffs, and alternatives.

Theory of operation. The general Ethernet approach uses a shared communications channel managed with a distributed control policy known as *carrier sense multiple access with collision detection*, or *CSMA/CD*. With this approach, there is no central controller managing access to the channel, and there is no fixed pre-allocation of time slots or fixed sharing of frequency bands. A station wishing to transmit is said to *contend* for use of the common shared communications channel (sometimes called the *Ether*) until it *acquires* the channel; once the channel is acquired the station uses it to transmit a packet.

To acquire the channel, stations check whether the network is busy (that is, use *carrier sense*) and *defer* transmission of their packet until the Ether is quiet (no other transmissions occurring). When quiet is detected, the deferring station immediately begins to transmit. During transmission, the transmitting station listens for a *collision* (other transmitters attempting to use the channel simultaneously). In a correctly functioning system, collisions may occur only within a short time interval following the start of transmission, since after this interval all stations will detect carrier and defer transmission. This time interval is called the *collision window* or the *collision interval*, and is a function of the end-to-end propagation delay. If no collisions occur during this time, a transmitter has acquired the Ether and continues transmission of the packet; but collision monitoring must still be done in case a malfunctioning station begins to transmit. If a station detects collision, the transmission of the rest of the packet is immediately aborted. To ensure that all parties to the collision have properly detected it, any station which detects a collision invokes a *collision consensus enforcement procedure* that briefly *jams* the channel. Each transmitter involved in the collision then schedules its packet for *retransmission* at some later time.

To minimize repeated collisions, each station involved in a collision tries to retransmit at a different time, by scheduling the retransmission to take place after a *random delay period.* In order to achieve channel stability under overload conditions, a controlled retransmission strategy is used: whereby the mean of the random retransmission delay is increased as a function of the channel load. An estimate of the channel load can be derived by monitoring the number of collisions experienced by any one packet. Among the options available for decentralized decision and control problems of this class, it has been shown that this strategy is optimal [Schoute, 1977].

Stations accept packets addressed to them, and discard any that are found to be in error. Deference reduces the probability of collision, and collision detection allows the timely retransmission of a packet. It is impossible, however, to guarantee that all packets transmitted will be delivered successfully; for example, if a receiver is not enabled, an error-free packet addressed to

it will not be delivered; higher-levels of protocol must detect these situations and retransmit.

Under very high load, short periods of time on the channel may be lost due to collisions, but the collision resolution procedure operates quickly [Metcalfe, 1973b; Metcalfe and Boggs, 1976; Shoch and Hupp, 1979, 1980a]. Channel utilization under these conditions will remain high, particularly if packets are large with respect to the collision interval. One of the fundamental parameters of any Ethernet implementation is the length of this collision interval, which is based on the propagation time between the furthest two points in the system.

Basic components. The CSMA/CD access procedure may use any broadcast multi-access channel (radio, twisted pair, coaxial cable, diffuse infrared, fiber optics [Rawson and Metcalfe, 1978], and others). Figure 1 illustrates a typical Ethernet system using coaxial cable. The four components are:

Station. A station or host makes use of the communication system, and is the basic addressable device connected to an Ethernet; in general it is a computer. We do not expect that "simple" terminals will be connected directly to an Ethernet; terminals can be connected to some form of terminal controller, however, which provides access to the network. In the future, as the level of sophistication in terminal increases, many terminals will support direct connection to the network. Furthermore, specialized I/O devices, such as magnetic tapes or disk drives, may incorporate sufficient computing resources to function as stations on the network.

Within the station there is some interface between the operating system environment and the Ethernet controller. The nature of this interface (often in software) depends upon the particular implementation of the controller functions in the station.

Controller. A controller for a station is really the set of functions and algorithms needed to manage access to the channel. These include signalling conventions, encoding and decoding, serial-to-parallel conversion, address recognition, error detection, buffering, the basic CSMA/CD channel management, and packetization. These functions can be grouped into two logically independent sections of the controller: the *transmitter* and the *receiver.*

The controller functions are generally implemented using some combination of hardware, microcode, and software, depending upon the nature of the station. It would be possible, for example, for a very capable station to have a minimal hardware connection to the transmission system, and perform most of these functions in software. Alternatively, a station might implement all the controller functions in hardware, or perhaps in a controller-specific microprocessor. Most controller implementations fall somewhere in between. With the continuing advances in LSI development, many of these functions will be packaged in a single chip, and several semiconductor manufacturers have already announced plans to build Ethernet controllers. The precise boundary between functions performed on the chip and those in the station is implementation-dependent, but the nature of that interface is of great importance. As many of the functions as possible should be moved into a chip, provided that this preserves all of the flexibility needed in the construction and

use of system interfaces and higher-level software.

The description of the controller in this paper is functional in nature: it indicates how the controller must behave independent of particular implementations. There is some flexibility in implementing a correct controller, and we make several recommendations that will contribute to efficient operation of the system.

Controller-to-transmission-system interface. One of the major interfaces in an Ethernet system is the point at which the controller in a station connects to the transmission system. The controller does much of the work in managing the communications process, so this is a fairly simple interface. It includes a path for data going to and from the transmission system. The data received can be used by the controller to sense carrier, but the transmission system normally includes a medium-specific mechanism for detecting collisions on the channel; this must also be communicated through the interface to the controller. It is possible to power a transceiver from a separate power source, but power is usually taken from the controller interface. In most transmission systems, the connection from the controller is made to a *transceiver*, and this interface is therefore called the *transceiver cable interface.*

Transmission system. The transmission system includes all the components used to establish a communications path among the controllers. In general, this includes a suitable broadcast *transmission medium*, the appropriate transmitting and receiving devices called *transceivers*, and, optionally, *repeaters* to extend the range of the medium. The protocol for managing access to the transmission system is implemented in the controller; the transmission system does not attempt to interpret any of the bits transmitted on the channel.

The broadcast transmission medium contains those components that provide a physical communication path. In the case of coaxial cable, this includes the cable plus any essential hardware—*connectors*, *terminators*, and *taps.*

Transceivers contain the electronics to transmit and receive signals on the channel. They recognize the presence of a signal when another station transmits, and recognize a collision that takes place when two or more stations transmit simultaneously.

Repeaters are used to extend the length of the transmission system beyond the physical limits imposed by the transmission medium. A repeater uses two transceivers to connect to two different Ethernet *segments* and combines them into one logical channel, amplifying and regenerating signals as they pass through in either direction [Boggs and Metcalfe, 1978]. Repeaters are transparent to the rest of the system, and stations on different segments can still collide. Thus, the repeater must propagate a collision detected on one segment through to the other segment, and it must do so without becoming unstable. A repeater makes an Ethernet channel longer, and as a result also increases the maximum propagation delay of the system—delay through the repeater and propagation delay through the additional segments. To avoid multipath interference in an Ethernet installation, there must be only one path between any two stations through the network. (The higher-level internetwork architecture may support alternate paths between stations through different

communications channels.)

Two generations of Ethernet designs. The Experimental Ethernet confirmed the feasibility of the design, and dozens of installations have been in regular use. These installations support hundreds of stations and a wide-ranging set of applications: file transfer, mail distribution, document printing, terminal access to hosts, data base access, copying disks, multi-machine programs, and more. Stations include the Alto workstation [Thacker, *et al.*, 1979], the Dorado (an internal research machine) [Dorado, 1981], the Digital Equipment PDP-11, and the Data General Nova. The system has been the subject of extensive performance measurements confirming its theoretically predicted behavior [Shoch and Hupp, 1979, 1980a].

Based upon that experience, a second generation system was designed at Xerox in the late 1970's. That effort subsequently led to the joint development of the Ethernet Specification. Stations built by Xerox for this network include the Xerox 860, the Xerox 8000 Network System Processor, and the Xerox 1100 Scientific Information Processor (the "Dolphin").

The two systems are very similar: they both use coaxial cable, Manchester signal encoding, CSMA/CD with dynamic control, etc. Some changes were made based either upon experience with the experimental system, or in an effort to enhance the characteristics of the network. Some of the differences between the two systems are summarized in the accompanying table.

	Experimental Ethernet	Ethernet Specification
Data rate	2.94 Mbps	10 Mbps
Maximum end-to-end length	1 Km	2.5 Km
Maximum segment length	1 Km	500 m
Encoding	Manchester	Manchester
Coax cable impedance	75 ohm	50 ohm
Coax cable signal levels	0 to +3V	0 to -2V
Transceiver cable connectors	25 and 15 pin D series	15 pin D series
Length of preamble	1 bit	64 bits
Length of CRC	16 bits	32 bits
Length of address fields	8 bits	48 bits

A "Concise Ethernet Specification," bringing together on two pages the important features of version 2 of the joint specification is included for reference. (Please note: in building a compatible device or component, the full Ethernet Specification [Ethernet, 1980, 1981] remains the controlling document.)

Figure 2 is a photograph of some typical components from the Experimental Ethernet, including a transceiver and tap, transceiver cable, and an Alto controller board. Figure 3 is a photograph of similar components based on the Ethernet Specification. Note that both controller boards have been implemented with standard MSI circuits.

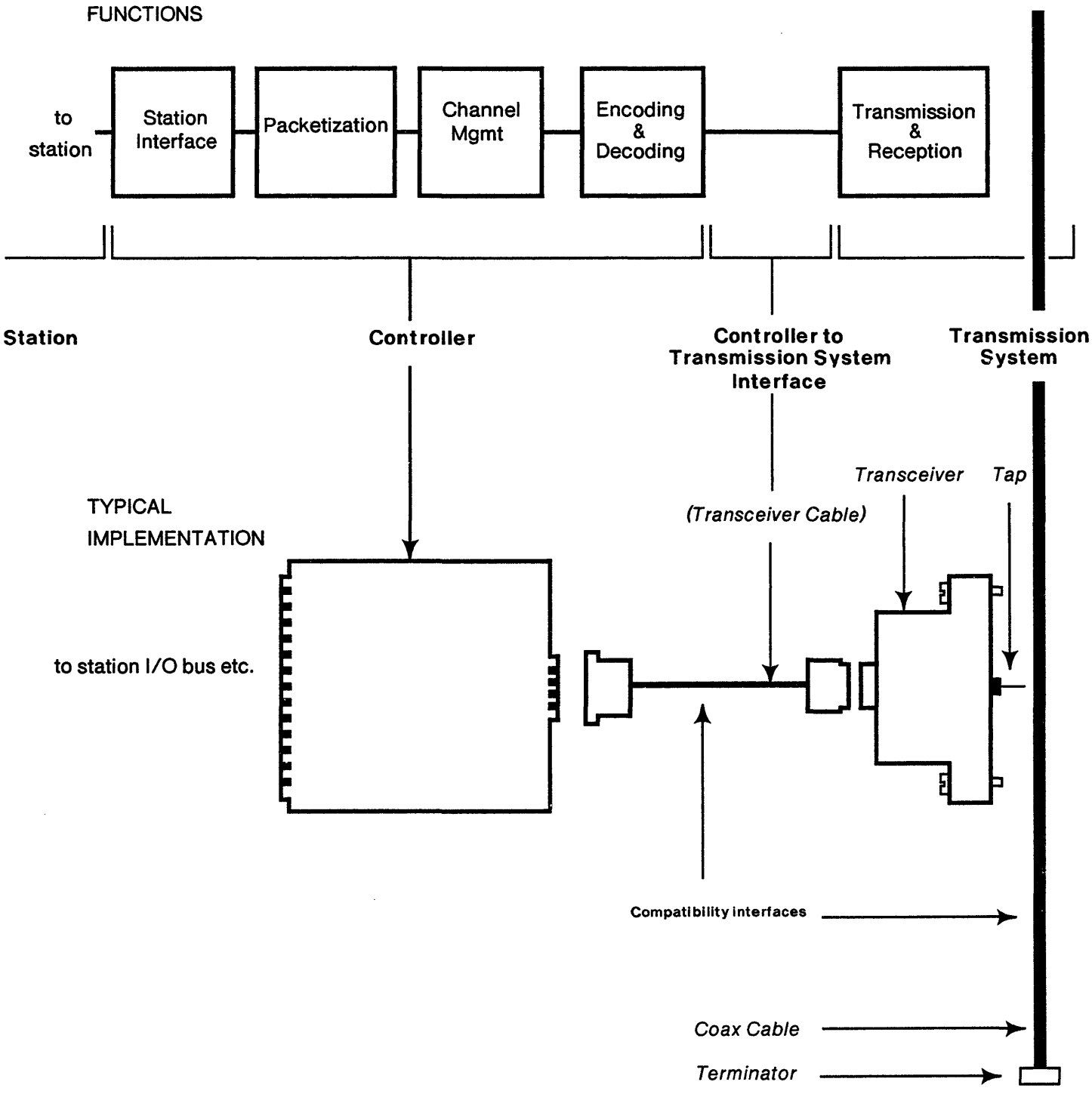
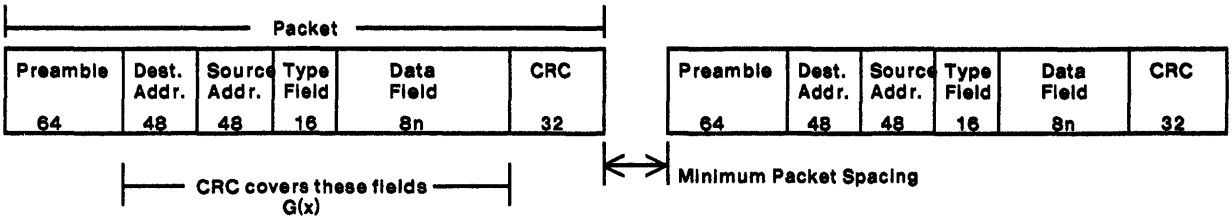


Figure 1 A General Ethernet Implementation

Concise Ethernet Specification

Packet Format



Stations must be able to transmit and receive packets on the common coaxial cable with the indicated packet format and spacing. Each packet should be viewed as a sequence of 8-bit bytes; the least significant bit of each byte (starting with the preamble) is transmitted first.

Maximum Packet Size: 1526 bytes (8 byte preamble + 14 byte header + 1500 data bytes + 4 byte CRC)

Minimum Packet Size: 72 bytes (8 byte preamble + 14 byte header + 46 data bytes + 4 byte CRC)

Preamble: This 64-bit synchronization pattern contains alternating 1's and 0's, ending with two consecutive 1's.
The preamble is: 10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011.

Destination Address: This 48-bit field specifies the station(s) to which the packet is being transmitted. Each station examines this field to determine whether it should accept the packet. The first bit transmitted indicates the type of address. If it is a 0, the field contains the unique address of the one destination station. If it is a 1, the field specifies a logical group of recipients; a special case is the broadcast (all stations) address, which is all 1's.

Source Address: This 48-bit field contains the unique address of the station that is transmitting the packet.

Type Field: This 16-bit field is used to identify the higher-level protocol type associated with the packet. It determines how the data field is interpreted

Data Field: This field contains an integral number of bytes ranging from 46 to 1500. (The minimum ensures that valid packets will be distinguishable from collision fragments.)

Packet Check Sequence: This 32-bit field contains a redundancy check (CRC) code, defined by the generating polynomial:

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

The CRC covers the address (destination/source), type, and data fields. The first transmitted bit of the destination field is the high-order term of the message polynomial to be divided by G(x) producing remainder R(x). The high-order term of R(x) is the first transmitted bit of the Packet Check Sequence field. The algorithm uses a linear feedback register which is initially preset to all 1's. After the last data bit is transmitted, the contents of this register (the remainder) are inverted and transmitted as the CRC field. After receiving a good packet, the receiver's shift register contains 11000111 00000100 11011101 01111011 (x^{31}, ... ,x^0).

Minimum Packet Spacing: This spacing is 9.6 usec, the minimum time that must elapse after one transmission before another transmission may begin.

Round-trip Delay: The maximum end-to-end, round-trip delay for a bit is 51.2 usec.

Collision Filtering: Any received bit sequence smaller than the minimum valid packet (with minimum data field) is discarded as a collision fragment.

Control Procedure

The control procedure defines how and when a host station may transmit packets into the common cable. The key purpose is fair resolution of occasional contention among transmitting stations.

Defer: A station must not transmit into the coaxial cable when carrier is present or within the minimum packet spacing time after carrier has ended.

Transmit: A station may transmit if it is not deferring. It may continue to transmit until either the end of the packet is reached or a collision is detected.

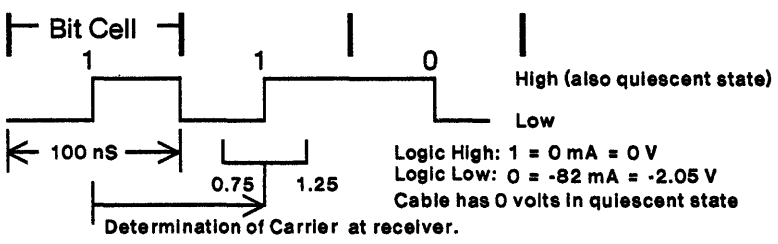
Abort: If a collision is detected, transmission of the packet must terminate, and a *jam* (4-6 bytes of arbitrary data) is transmitted to ensure that all other participants in the collision also recognize its occurrence.

Retransmit: After a station has detected a collision and aborted, it must wait for a random *retransmission delay*, defer as usual, and then attempt to retransmit the packet. The random time interval is computed using the backoff algorithm (below). After 16 transmission attempts, a higher level (e.g. software) decision is made to determine whether to continue or abandon the effort.

Backoff: Retransmission delays are computed using the *Truncated Binary Exponential Backoff* algorithm, with the aim of fairly resolving contention among up to 1024 stations. The delay (the number of time units) before the n^{th} attempt is a uniformly distributed random number from [0 to 2^n-1] for $0 < n \leq 10$ (n=0 is the original attempt). For attempts 11-15, the interval is *truncated* and remains at [0 to 1023]. The unit of time for the retransmission delay is 512 bit times (51.2 usec).

Channel Encoding

Manchester encoding is used on the coaxial cable. It has a 50% duty cycle, and insures a transition in the middle of every bit cell ("data transition"). The first half of the bit cell contains the complement of the bit value, and the second half contains the true value of the bit.

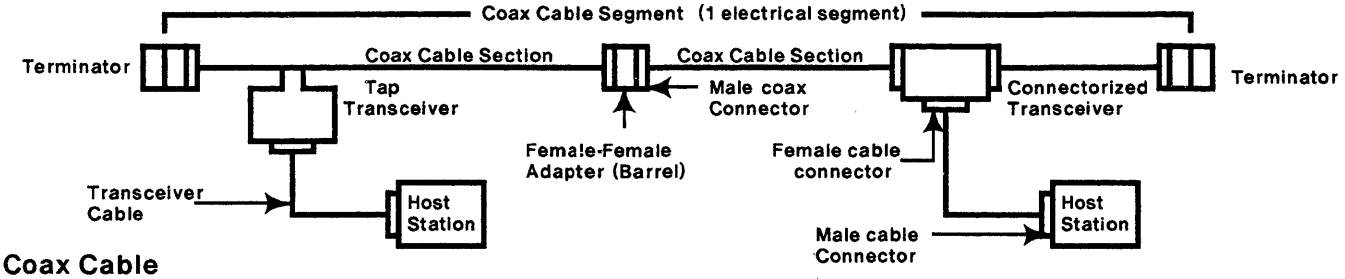


Data Rate

Data rate is 10 M bits/sec = 100 nsec bit cell ± 0.01%.

Carrier

The presence of data transitions indicates that carrier is present. If a transition is not seen between 0.75 and 1.25 bit times since the center of the last bit cell, then carrier has been lost, indicating the end of a packet. For purposes of deferring, carrier means any activity on the cable, independent of being properly formed. Specifically, it is any activity on either receive or collision detect signals in the last 160 nsec.



Coax Cable

Impedance: 50 ohms ± 2 ohms (Mil Std. C17-E). This impedance variation includes batch-to-batch variations. Periodic variations in impedance of up to ± 3 ohms are permitted along a single piece of cable.

Cable Loss: The maximum loss from one end of a cable segment to the other end is 8.5 db at 10 MHz (equivalent to ~500 meters of low loss cable).

Shielding: The physical channel hardware must operate in an ambient field of 2 volts per meter from 10 KHz to 30 MHz and 5 V/meter from 30 MHz to 1 GHz. The shield has a transfer impedance of less than 1 milliohm per meter over the frequency range of 0.1 MHz to 20 MHz (exact value is a function of frequency).

Ground Connections: The coax cable shield shall not be connected to any building or AC ground along its length. If for safety reasons a ground connection of the shield is necessary, it must be in only one place.

Physical Dimensions: This specifies the dimensions of a cable which can be used with the *standard tap*. Other cables may also be used, if they are not to be used with a tap-type transceiver (such as use with connectorized transceivers, or as a section between sections to which standard taps are connected).

Center Conductor:	0.0855" diameter solid tinned copper
Core Material:	Foam polyethylene or foam teflon FEP
Core O.D.:	0.242 " minimum
Shield:	0.326" maximum shield O.D. (>90% coverage for outer braid shield)
Jacket:	PVC or teflon FEP
Jacket O.D.:	0.405"

Coax Connectors and Terminators

Coax cables must be terminated with male N-series connectors, and cable sections will be joined with female-female adapters. Connector shells shall be insulated such that the coax shield is protected from contact to building grounds. A sleeve or boot is acceptable. Cable segments should be terminated with a a female N-series connector (can be made up of a barrel connector and a male terminator) having an impedance of 50 ohms ± 1%, and able to dissipate 1 watt. The outside surface of the terminator should also be insulated.

Transceiver

CONNECTION RULES

Up to 100 transceivers may be placed on a cable segment no closer together than 2.5 meters. Following this placement rule reduces to a very low (but not zero) probability the chance that objectionable standing waves will result.

COAX CABLE INTERFACE

Input Impedance: The resistive component of the impedance must be greater then 50 Kohms. The total capacitance must be less than 4 picofarads.

Nominal Transmit Level: The important parameter is average DC level with 50% duty cycle waveform input. It must be -1.025 V (41 mA) nominal with a range of -0.9 V to -1.2 V (36 to 48 mA). The peak-to-peak AC waveform must be centered on the average DC level and its value can range from 1.4 V P-P to twice the average DC level. The voltage must never go positive on the coax. The quiescent state of the coax is logic high (0 V). Voltage measurements are made on the coax near the transceiver with the shield as reference. Positive current is current flowing out of the center conductor of the coax.

Rise and Fall Time: 25 nSec ± 5 nSec with a maximum of 1 nSec difference between rise time and fall time in a given unit. The intent is that dV/dt should not significantly exceed that present in a 10 MHz sine wave of same peak-to-peak amplitude.

Signal Symmetry: Asymmetry on output should not exceed 2 nSec for a 50-50 square wave input to either transmit or receive section of transceiver.

TRANSCEIVER CABLE INTERFACE

Signal Pairs: Both transceiver and host station shall drive and present at the receiving end a 78 ohm balanced load. The differential signal voltage shall be 0.7 volts nominal peak with a common mode voltage between 0 and +5 volts using power return as reference. (This amounts to shifted ECL levels operating between Gnd and +5 volts. A 10116 with suitable pulldown resistor may be used). The quiescent state of a line corresponds to logic high, which occurs when the + line is more positive than the - line of a pair.

Collision Signal: The active state of this line is a 10 MHz waveform and its quiescent state is logic high. It is active if the transceiver is transmitting and another transmission is detected, or if two or more other stations are transmitting, independent of the state of the local transmit signal.

Power: +11.4 volts to +16 volts DC at controller. Maximum current available to transceiver is 0.5 ampere. Actual voltage at transceiver is determined by the interface cable resistance (max 4 ohms loop resistance) and current drain.

ISOLATION

The impedance between the coax connection and the transceiver cable connection must exceed 250 Kohms at 60 Hz and withstand 250 VRMS at 60 Hz.

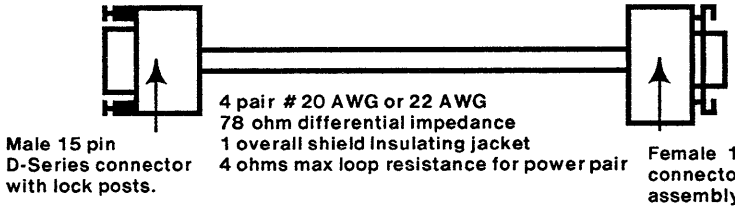
Transceiver Cable and Connectors

Maximum signal loss = 3 db @ 10 MHz. (equivalent to ~50 meters of either 20 or 22 AWG twisted pair).

Transceiver Cable Connector Pin Assignment

1.	Shield*		
2.	Collision +	9.	Collision -
3.	Transmit +	10.	Transmit -
4.	Reserved	11.	Reserved
5.	Receive +	12.	Receive -
6.	Power Return	13.	+ Power
7.	Reserved	14.	Reserved
8.	Reserved	15.	Reserved

*Shield must be terminated to connector shell.



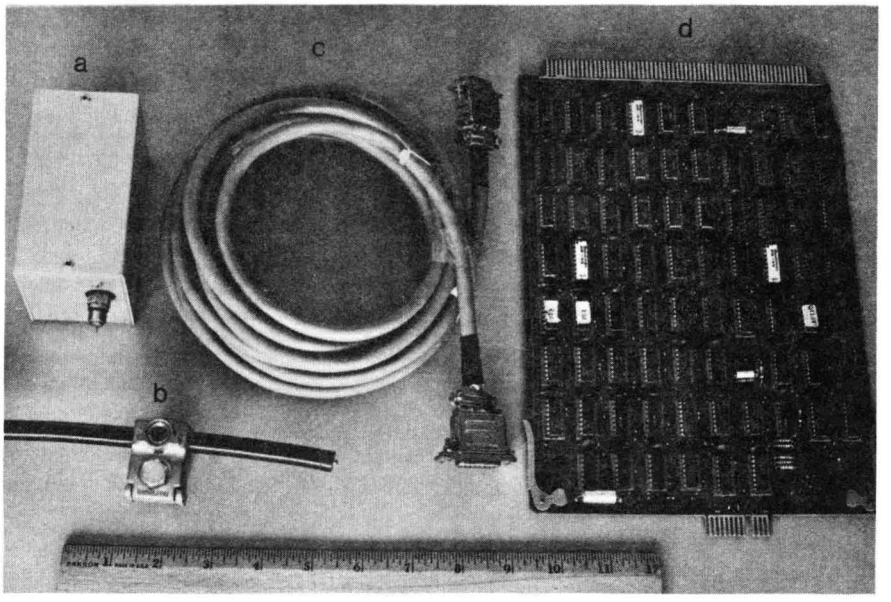


Figure 2. Experimental Ethernet Components: (a) transceiver and tap, (b) tap-block, (c) transceiver cable, and (d) Alto controller board.

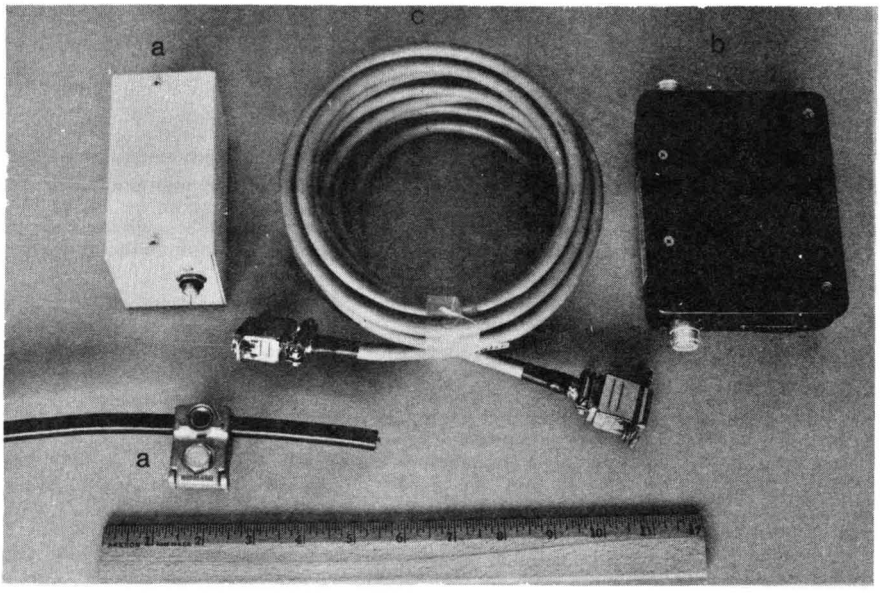


Figure 3. Ethernet Specification Components: (a) transceiver, tap, and tap-block, (b) connectorized transceiver, (c) transceiver cable.

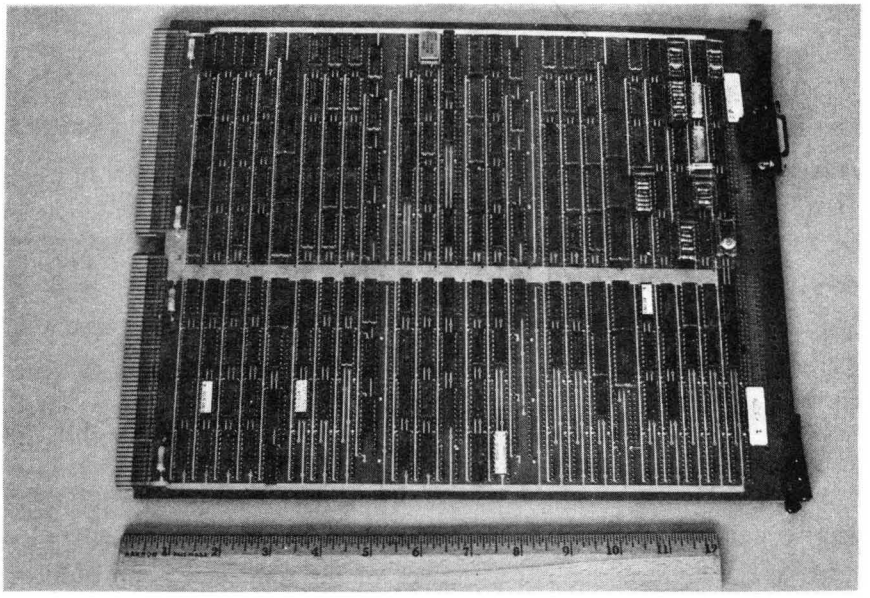


Figure 3. Ethernet Specification Components: (d) Dolphin controller board.

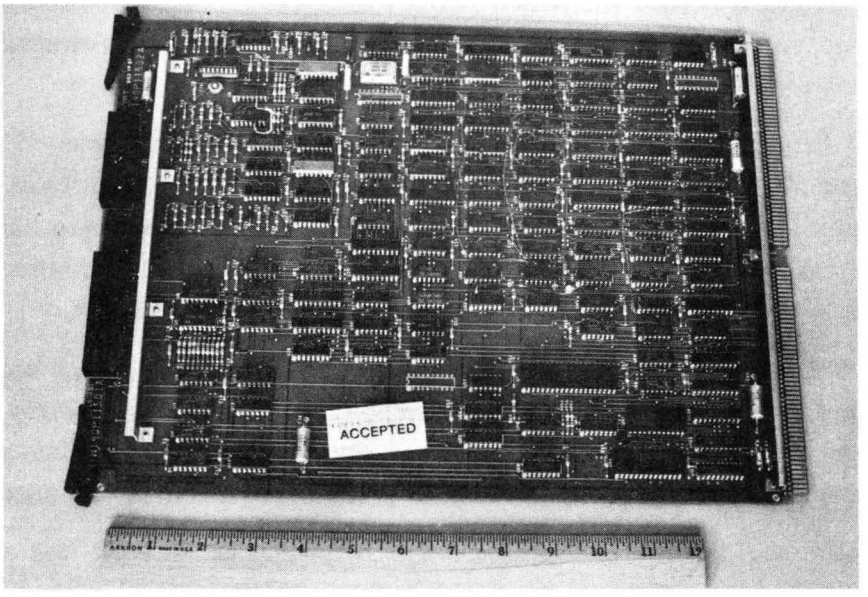


Figure 3. Ethernet Specification Components: (e) Xerox 8000 controller board (half the board).

3. Transmission System Design

We now turn to the main topic: the design issues and tradeoffs that emerged in the development of the Ethernet local computer network, and the lessons learned from that experience. This section deals with the transmission system.

Coaxial cable subsystem. In addition to having favorable signalling characteristics and the ability to handle multi-megabit transmission speeds, a single coaxial cable can support communication among many different stations. The mechanical aspects of coaxial cable make it feasible to tap in at any point without severing the cable or producing excessive RF leakage; features having to do with installation, maintenance, and reconfigurability are important considerations in any local network design.

There are reflections and attenuation in a cable, however, and these combine to impose some limits on the system design. Engineering the shared channel entails tradeoffs among: data rate on the cable, length of the cable, electrical characteristics of the transceiver, and number of stations. For example, it is possible to operate at very high data rates over short distances, but the rate must be reduced to support a greater maximum length. Also, if each transceiver introduces significant reflections it may be necessary to limit the placement and possibly the number of transceivers.

The characteristics of the coaxial cable fix the maximum data rate, but the actual clock is generated in the controller. Thus, the station interface and controller must be designed to match the data rates used over the cable. Selection of coaxial cable as the transmission medium has no other direct impact on either the station or the controller.

Cable. The Experimental Ethernet used 75 ohm, RG-11 type foam cable. The Ethernet Specification uses a 50 ohm, solid center conductor, double shield, foam dielectric cable, in order to provide some reduction in the magnitude of reflections from insertion capacitance (introduced by tapping into the cable), and to provide better immunity to environmental electromagnetic noise. Belden AWM Style 1478 Ethernet Coax JE, meets the Ethernet Specification.

Terminators and connectors. A small terminator is attached to the cable at each end to provide a termination impedance for the cable, equal to its characteristic impedance, thereby eliminating reflection from the ends of the cable. For convenience, the cable can be divided into a number of *sections*, using simple connectors between sections, to produce one electrically continuous *segment.*

Segment length and the use of repeaters. The Experimental Ethernet was designed to accommodate a maximum end-to-end length of 1 Km, implemented as a single electrically continuous segment. Active repeaters could be used with that system to create complex topologies that would cover a wider area in a building (or complex of buildings) within the end-to-end length limit. With the use of those repeaters, however, the maximum end-to-end length between any two stations was still meant to be approximately 1 Km. Thus, the segment length and the maximum end-to-end length were the same, and repeaters were used to provide additional flexibility.

In developing the Ethernet Specification the strong desire to support a 10 Mbps data rate—with reasonable transceiver cost—led to a maximum segment length of 500 meters. We expect that this length will be sufficient to support many installations and applications with a single Ethernet segment. In some cases, however, we recognized a requirement for greater maximum end-to-end length in one network. In these cases, repeaters may now be used not just for additional flexibility, but also to extend the overall length of an Ethernet. The Ethernet Specification permits the concatenation of up to three segments; the maximum end-to-end delay between two stations measured as "a distance" is 2.5 Km (this includes the delay through repeaters containing a point-to-point link [Ethernet, 1980]).

Taps. Transceivers may connect to a coax cable with the use of a *pressure tap*, borrowed from CATV technology. Such a tap allows connection to the cable without cutting it to insert a connector, and avoids the need to interrupt network service while installing a new station. One design uses a *tap-block* that is clamped on the cable and uses a special tool to penetrate the outer jacket and shield. The tool is removed and the separate tap is screwed into the block. Another design has the tap and tap-block integrated into one unit, with the tap puncturing the cable to make contact with the center conductor as the tap-block is being clamped on.

Alternatively, the cable can be cut, and connectors fastened to each piece of cable. This unfortunately disrupts the network during the installation process. After the connectors are installed at the break in the cable, a *T-connector* can be inserted in between, and then connected to a transceiver. As another option, a *connectorized transceiver* has two connectors built into it for direct attachment to the cable ends without a T-connector.

Experimental Ethernet installations have used pressure taps, where the tap and tap-block are separate, as illustrated in Figure 2. Installations conforming to the Ethernet Specification have used all the options; Figure 3 illustrates a connectorized transceiver, and a pressure tap with separate tap and tap-block.

Transceiver. The transceiver is the most important part of the transmission system. It couples the station to the cable.

The controller-to-transmission system interface is very simple, and functionally has not changed between the two Ethernet designs. It performs four functions: (1) transfer transmit data from the controller to the transmission system, (2) transfer receive data from the transmission system to the controller, (3) indicate to the controller that a collision is taking place, and (4) provide power to the transmission system.

It is important that the two ground references in the system—the common coaxial cable shield, and the local ground associated with each station—not be tied together, since one local ground typically may differ from another local ground by many volts. Connection of several local grounds to the common cable could cause a large current to flow through the cable's shield, introducing noise and presenting a potential safety hazard.

It is the transceiver which provides this ground isolation between signals from the controller and signals on the cable. Several isolation techniques are possible: transformer isolation, optical isolation, and capacitive isolation. Transformer isolation provides both power and signal isolation; it has low differential impedance for signals and power, and a high common mode impedance for isolation. It is also relatively inexpensive to implement. Optical isolators that preserve tight signal symmetry at a competitive price are not readily available. Capacitive coupling is inexpensive and preserves signal symmetry, but has poor common mode rejection. It is for these reasons that transformer isolation is used in Ethernet Specification transceivers. In addition, the mechanical design and installation of the transceiver must preserve this isolation. For example, cable shield connections should not come in contact with a building ground (e.g., a cable tray, conduit, or ceiling hanger).

The transceiver provides a high-impedance connection to the cable in both the power-on and power-off states. In addition, it should protect the network from possible internal circuit failures that could cause it to disrupt the network as a whole. It is also important for the transceiver to withstand transient voltages on the coax between the center conductor and shield. While this should not occur if the coax shield is grounded in only one place, such isolation may not exist during installation [Crane and Taft, 1980].

Negative transmit levels were selected for the Ethernet Specification to permit use of fast NPN transistors (which are more easily integrated) for the output current source. A current source output was chosen over the voltage source used in the Experimental Ethernet to facilitate collision detection (see discussion in the next section).

The key factor affecting the maximum number of transceivers on a segment in the Ethernet Specification is the input bias current for the transceivers. With easily achievable bias currents and collision threshold tolerances, the maximum number was conservatively set at 100 per segment. If the only factors taken into consideration were signal attenuation and reflections, then the number would have been larger.

4. Controller Design

The transmitter and receiver sections of the controller perform signal conversion, encoding and decoding, serial to parallel conversion, address recognition, error detection, CSMA/CD channel management, buffering and packetization. We postpone discussion of buffering and packetization until the section on controller-to-station interface design. This section deals with the various functions that the controller needs to perform, and the next section shows how they are coordinated into an effective CSMA/CD channel management policy.

Signalling, data rate, and framing. The transmitter generates the serial bit stream inserted into the transmission system. Clock and data are combined into one signal with the use of a suitable encoding scheme. Manchester encoding was used in the Experimental Ethernet because of its simplicity. In Manchester encoding, each bit cell has two parts: the first half of the cell is the complement of the bit value, and the second half of the cell is the bit value. Thus, there is always a transition in the middle of every bit cell, and this is used by the receiver to extract the data.

For the Ethernet Specification, MFM encoding (used in double density disk recording) was considered, but was rejected because decoding was more sensitive to phase distortions from the transmission system and required more components to implement. Compensation is not as easy as in the disk situation because a station must receive signals from both nearby and distant stations.

In the Experimental Ethernet a data rate anywhere in the range of 1 to 5 Mbps might have been chosen. The particular rate of 2.94 Mbps was convenient for working with the first Altos. For the Ethernet Specification, we wanted a data rate as high as possible; very high data rates, however, limit the effective length of the system, and require more-precise electronics. The data rate of 10 Mbps represents a tradeoff among these considerations.

Packet framing on the Ethernet is simple. The presence of a packet is indicated by the presence of *carrier*, or transitions. In addition, all packets begin with a known pattern of bits called the *preamble*. This is used by the receiver, first to establish bit synchronization, and then to locate the first bit of the packet. The preamble is inserted by the controller at the sending station, and stripped off by the controller at the receiving station. Packets may be of variable length and absence of carrier marks the end of a packet. Hence, there is no need to have framing flags and "bit stuffing" in the packet, as in other data link protocols like SDLC or HDLC.

The Experimental Ethernet used a 1-bit preamble. While this worked very well, we have, on rare occasions, seen some receivers that could not synchronize with this very short preamble [Shoch, in press]. The Ethernet Specification uses a 64-bit preamble to ensure synchronization of phase-lock loop receivers often used at the higher data rate. It is necessary to specify 64 bits to allow for (1) worst case tolerances on phase-lock loop components, (2) maximum times to reach steady state conditions through transceivers, and (3) loss of preamble bits owing to squelch on input and output within the transceivers. Note that the presence of a repeater can add up to four extra transceivers between a source and destination.

Additional conventions can be imposed upon the frame structure. Requiring that all packets be a multiple of some particular byte or word size simplifies controller design, and provides an additional consistency check. All packets on the Experimental Ethernet are viewed as a sequence of 16-bit words with the most significant bit of each word transmitted first. The Ethernet Specification requires all packets to be an integral number of 8-bit bytes (exclusive of the preamble, of course), with the least significant bit of each byte transmitted first. The order in which the bytes of an Ethernet packet are stored in the memory of a particular station is part of the controller-to-station interface (see Section 6).

Encoding and decoding. The transmitter is responsible for taking a serial bit stream from the station and encoding it into the Manchester format. The receiver is responsible for decoding an incoming signal and converting it into a serial bit stream for the station. The process of encoding is fairly straightforward, but that of decoding is more difficult, and is realized in a *phase decoder.* The known preamble pattern can be used to help initialize the phase decoder, which can employ any of several techniques including: an analog timing circuit, a phase-locked loop, or a digital phase decoder (which rapidly samples the input and does a pattern match). The particular decoding technique selected may be a function of the data rate, since some decoder designs may not run as fast as others. Some phase decoding techniques—particularly the digital one—have the added advantage of recognizing certain phase violations as collisions on the transmission medium. This is one way to implement collision detection, but may not generalize to all controllers (see the discussion of collision detection below).

The phase decoders used by stations on the Experimental Ethernet included: an analog timing circuit in the form of a delay line on the PDP-11; an analog timing circuit in the form of a simple one-shot-based timer on the Alto; and a digital decoder on the Dorado. All stations built by Xerox for the 10 Mbps Ethernet use phase-locked loops.

Carrier sense. Recognizing packets passing by is one of the important requirements of the Ethernet access procedure. Although transmission is baseband, we have borrowed the term "sensing carrier" from radio terminology, to describe the detection of signals on the channel. Carrier sense is used for two purposes: (1) in the receiver, to delimit the beginning and end of the packet, and (2) in the transmitter, to tell when it is permissible to send. With the use of Manchester phase encoding, carrier is conveniently indicated by the presence of transitions on the channel. Thus, the basic phase decoding mechanism can produce a signal indicating the presence of carrier, independent of the data being extracted. The Ethernet Specification requires a slightly subtle carrier sense technique owing to the possibility of a "saturated collision" (see below).

Collision detection. The ability to detect collisions and shut down the transmitter promptly is an important feature in minimizing the time on the channel lost to collisions. The general requirement is that, while transmitting, a controller must recognize that another station is also transmitting. There are two approaches:

1) Collision detection in the transmission system. It is usually possible for the transmission system itself to recognize a collision. This allows any medium-dependent technique to be used, and is usually implemented by comparing the injected signal with the received signal. Comparing the transmitted and received signals is best done in the transceiver where there is a known relationship between the two signals. It is the controller, however, which needs to know that a collision is taking place.

2) Collision detection in the controller. Alternatively, the controller itself can recognize a collision by comparing the transmitted signal with the received signal, or unilaterally attempting to recognize collisions, since they often appear as phase violations.

Both generations of Ethernet detect collision within the transceiver, and generate the collision signal in the controller-to-transmission-system interface. Where feasible, this can be supplemented with a collision detection process in the controller. Collision detection may not be absolutely foolproof. Some transmission schemes can recognize all collisions, but other combinations of transmission scheme and collision detection may not provide 100% recognition. For example, the Experimental Ethernet system functions, in principle, as a wired-OR. It is remotely possible that while one station transmits, another station sends a packet whose waveform, at the first station, exactly matches the signal sent by the first station, and thus no collision is recognized there. Unfortunately, the intended recipient might be between the two stations, and the two signals would indeed interfere.

There is another possible scenario in which collision detection breaks down. One station begins transmitting, and its signal propagates down the channel. Another station still senses the channel idle, begins to transmit, gets out a bit or two, and then detects a collision. If the colliding station shuts down immediately, it leaves a very small collision moving through the channel. In some approaches (e.g., DC threshold collision detection) this may be attenuated, and simply not make it back to the transmitting station and trigger its collision detection circuitry.

The probability of such events happening is small. Actual measurements on the Experimental Ethernet system indicate that the collision detection mechanism works very well. Yet it is important to remember that an Ethernet system only delivers packets with high probability, and not certainty.

To help ensure proper detection of collisions, each transmitter adopts a *collision consensus enforcement* procedure. This makes sure that all other parties to the collision will recognize that a collision has taken place. In spite of its lengthy name, this is a simple procedure: after detecting a collision, a controller transmits a *jam* that every operating transmitter should detect as a collision. In the Experimental Ethernet the jam is a phase violation, while in the Ethernet Specification it is the transmission of between 4 to 6 bytes of (random) data.

Another possible collision scenario arises in the context of the Ethernet Specification; it is possible that a collision involves so many participants that a transceiver is incapable of injecting any more current into the cable. During such a collision, one cannot guarantee that the waveform on the cable will exhibit any transitions (in the extreme case, it simply sits at a constant DC level equal to the saturation voltage). This is called a *saturated collision.* In this situation, the simple notion of

sensing carrier by detecting transitions would not work anymore. In particular, a station that deferred only when seeing transitions would think the Ether was idle and jump right in, becoming another participant in the collision. Of course it would immediately detect the collision and back off, but in the extreme case (everyone wanting to transmit), such jumping-in could (theoretically) cause the saturated collision to snowball and go on for a very long time. While we recognized that this form of instability was highly unlikely to occur in practice, we included a simple mechanism to aid sensing carrier in the Ethernet Specification, thereby preventing the problem.

This discussion has focused on collision detection by the transmitter of a packet. We have seen that the transmitter may depend on a collision detect signal generated unilaterally by its receiving phase decoder. Can this *receiver-based collision detection* be used just by a receiver (that is, a station which is not trying to transmit)? A receiver with this capability could immediately abort an input operation, and could even generate a jam signal to help ensure that the collision came to a prompt termination. With a reasonable transmitter-based collision detection scheme, however, the collision is recognized by the transmitters and the damaged packet would come to an end very shortly. Receiver-based collision detection could provide early warning of a collision for use by the receiver, but this is not a necessary function and we have not used it in either generation of Ethernet design.

CRC generation and checking. The transmitter generates a *cyclic redundancy check* (CRC) of each transmitted packet and appends it to a packet before transmission. The receiver checks the CRC on packets it receives, and strips it off before giving the packet to the station. If the CRC is incorrect there are two options: discard the packet, or deliver the damaged packet with an appropriate status indicating a CRC error.

While most CRC algorithms are quite good, they are not infallible. There is a small probability that undetected errors may slip through. More importantly, the CRC only protects a packet from the point at which the CRC is generated to the point at which it is checked. Thus, the CRC cannot protect a packet from damage that occurs in parts of the controller (for example, a FIFO in the parallel path to the memory of a station (the DMA), or in the memory itself). If error detection at a higher level is required, then an end-to-end software checksum can be added to the protocol architecture.

In measuring the Experimental Ethernet system, we have seen packets whose CRC was reported as correct, but whose software checksum was incorrect [Shoch, in press]. These did not necessarily represent an undetected Ethernet error, but usually resulted from an external malfunction—a broken interface, a bad CRC checker, or even an incorrect software checksum algorithm.

Selection of the CRC algorithm is guided by several concerns. It should have sufficient strength to properly detect virtually all packet errors. Unfortunately, only a limited set of CRC algorithms are currently implemented in LSI chips. The Experimental Ethernet used a 16-bit CRC, taking advantage of a single-chip CRC generator/checker. The Ethernet Specification provides better error detection by using a 32-bit CRC [Hammond, *et. al.*, 1975; Bittel, 1977]. This function will be easily

implemented in an Ethernet chip.

Addressing. The packet format includes both a source and destination address. A local network design may adopt either of two basic addressing structures: *network specific* host addresses or *unique* host addresses [Shoch, 1978]. In the first case, stations are assigned network addresses which must be unique on their network, but which may be the same as the address held by a station on another network. Such addresses are sometimes called *network-relative addresses*, since they depend upon the particular network to which the station is attached. In the second case, each station is assigned an address which is unique over all space and time. Such addresses are also known as *absolute* or *universal* addresses, drawn from a *flat address space.*

To permit internetwork communication, the network-specific address of a station must usually be combined with a unique network number in order to produce an unambiguous address at the next level of protocol. On the other hand, there is no need to combine an absolute station address with a unique network number to produce an unambiguous address. However, it is possible that internetwork systems based on flat (internetwork and local network), absolute addresses will include a unique network number at the internetwork layer as a "very strong hint" for the routing machinery.

If network-specific addressing is adopted, Ethernet address fields need only be large enough to accommodate the maximum number of stations that will be connected to one local network. In addition, there must be a suitable administrative procedure for assigning addresses to stations. Many installations will have more than one Ethernet, and if a station is moved from one network to another it may be necessary to change its network-specific address, since its former address may be in use on the new network. This was the approach used on the Experimental Ethernet, with an 8-bit field for the source and the destination addresses.

We anticipate that there will be a large number of stations and many (local) networks in an internetwork. Thus, the management of network-specific station addresses can represent a severe problem. The use of a flat address space provides for reliable and manageable operation as a system grows, as machines move, and as the overall topology changes. A flat internet address space requires that the address space be large enough to ensure uniqueness and provide adequate room for growth. It is most convenient if the local network can directly support these fairly large address fields.

For these reasons the Ethernet Specification uses 48-bit addresses. Note that these are station addresses, and are not associated with a particular network interface or controller. In particular, we believe that higher level routing and addressing procedures are simplified if a station connected to multiple networks has only one identity which is unique over all networks. The address should not be "hard-wired" into a particular interface or controller, but should be settable from the station. It may be very useful, however, to allow a station to read a unique host identifier from the controller. The station can then choose whether to return this identifier to the controller as its address.

In addition to addressing a single station, several enhanced addressing modes also are desirable. *Multicast* addressing is a mechanism by which packets may be targeted to more than one destination. This kind of service is particularly valuable in certain kinds of distributed applications, such as the access and update of distributed data bases, teleconferencing, and the distributed algorithms which are used to manage the network (and the internetwork). We believe that multicast should be supported by allowing the destination address to specify either a physical or logical address. A logical address is known as a *multicast ID*. *Broadcast* is a special case of multicast where a packet is intended for all active stations. Both generations of Ethernets support broadcast, while only the Ethernet Specification directly supports multicast. A detailed discussion of 48-bit addresses can be found in [Dalal and Printis, 1981]

Stations supporting multicast must filter multicast IDs of interest. Because of the anticipated wide growth in the use of multicast service, serious consideration should be given to the aspects of the station and controller design which reduce the system load required to filter unwanted multicast IDs. Broadcast should be used with discretion since all nodes incur the overhead of processing every broadcast packet.

Controllers capable of accepting packets regardless of destination address provide *promiscuous* address recognition. On such stations, one can develop software to observe all of the channel's traffic and construct traffic matrices, perform load analysis, potentially perform fault isolation, and debug protocol implementations. While such a station is able to read packets not addressed to it, we expect that sensitive data will be encrypted by higher levels of software.

5. CSMA/CD Channel Management

A major portion of the controller is devoted to Ethernet channel management. These conventions specify procedures by which packets are transmitted and received on the multi-access channel.

Transmitter. The transmitter is invoked when the station has a packet to send. If a collision occurs, the controller enforces the collision with a suitable jam, shuts down the transmitter, and schedules a retransmission.

Retransmission policies have two conflicting goals: (1) scheduling a retransmission quickly, to get the packet out and maintain use of the channel, and (2) voluntarily "backing off" to reduce the station's load on a busy channel. Both generations of Ethernets use the *binary exponential backoff algorithm*, described below. After some maximum number of collisions the transmitter gives up, and reports a suitable error back to the station—both generations of Ethernets give up after 16 collisions.

The binary exponential backoff algorithm is used to calculate the *delay* before retransmission. After a collision takes place the objective is to obtain delay periods that will reschedule each station at points in time quantized in steps at least as large as a collision interval. This time quantization is called the *retransmission slot time.* To guarantee quick use of the channel, this slot time should be short, but to avoid collisions it should be larger than a collision interval. Therefore, the slot time is usually set to be a little longer than the round-trip time of the channel. The retransmission delay is computed as the product of some *retransmission count* (a positive integer) and the retransmission slot time.

To minimize the probability of repeated collisions, each retransmission count is selected as a random number from a particular *retransmission interval*, between zero and some upper limit. In order to control the channel and keep it stable under high load, the interval is doubled with each successive collision, thus extending the range of possible retransmission delays. This algorithm has very short retransmission periods at the beginning, but will back off quickly, preventing the channel from becoming overloaded. After some number of backoffs, the retransmission interval becomes large. To avoid undue delays and slow response to improved channel characteristics, the doubling can be stopped at some point, with additional retransmissions still being drawn from this interval, before the transmission is finally aborted. This is referred to as *truncated binary exponential backoff.*

The truncated binary exponential backoff algorithm approximates the ideal algorithm where the probability of transmission of a packet is $1/Q$, when the number of stations attempting to transmit is Q [Metcalfe, 1973a]. The retransmission interval is truncated when Q becomes equal to the maximum number of stations.

In the Experimental Ethernet, the very first transmission attempt proceeds with no delay (that is, the retransmission interval is [0-0]). The retransmission interval is doubled after each of the first

eight transmission attempts. Thus, the retransmission count should be uniformly distributed between 0 and $2^{\min(\text{retransmission attempt},8)} - 1$. After the first transmission attempt, the next eight intervals will be [0-1], [0-3], [0-7], [0-15], [0-31], [0-63], [0-127], and [0-255]. The retransmission interval remains at [0-255] on any subsequent attempts as the maximum number of stations is 256. The Ethernet Specification has the same algorithm with ten intervals, since the network permits about 1000 stations; the maximum interval is therefore [0-1023]. The backoff algorithm restarts with a zero retransmission interval for the transmission of every new packet.

This particular algorithm was chosen because it has the proper basic behavior, and because it allows a very simple implementation. The algorithm is now supported by empirical data verifying the stability of the system under heavy load [Shoch & Hupp, 1979, 1980a]. Additional attempts made to explore more sophisticated algorithms resulted in negligible performance improvement.

Receiver. The receiver is invoked in the controller when carrier appears on the channel. The controller processes the incoming bit stream in the following manner:

The remaining preamble is first removed. If the bit stream ends before the preamble completes, the bit stream was probably the result of a short collision, and the receiver is restarted.

The receiver next determines whether the packet is addressed to it or not. The controller will accept a packet in one of several circumstances: (1) the destination address matches the specific address of the station, (2) the destination address has the distinguished broadcast destination, (3) the destination address is a multicast group of which the station is a member, or (4) the station has set the controller in promiscuous mode and receives all packets. Some controller designs might choose to receive the entire packet before invoking the address recognition procedure. That is feasible, but consumes both memory and processing resources in the controller. More typically, address recognition takes place at a fairly low level in the controller, and if the packet is not to be accepted, the controller can ignore the rest of it.

The receiver now accepts the entire packet. Before delivering the packet to the station, the CRC is verified and other consistency checks are performed. For example, the packet should typically end on an appropriate byte or word boundary, and be of appropriate minimum length; a minimum packet would have to include at least a destination and source address, a packet type, and a CRC. Collisions on the channel, however, can produce short, damaged packets (called *collision fragments*, and described below). It is generally unnecessary to report these errors to the station, since they can be eliminated with a *fragment filter* in the controller. It is important, however, for the receiver to be restarted promptly after a collision fragment is received, since the sender of the packet may be about to retransmit.

Packet length. One important goal of the Ethernet is "data transparency." In principle, this means that the data field of a packet may contain any bit pattern, and may be of any length, from zero to arbitrarily large. In practice, while it is easy to allow any bit pattern to appear in the data field, there are some practical considerations that suggest imposing upper and lower bounds on its length.

At one extreme, an *empty* packet (one with a zero-length data field) would consist of just a preamble, source and destination addresses, a type field, and a CRC. The Experimental Ethernet permitted empty packets. However, in some situations (described below) it is desirable to enforce a minimum overall packet size by mandating a minimum-length data field, as in the Ethernet Specification; higher-level protocols wishing to transmit shorter packets must then pad out the data field to reach the minimum.

At the other extreme, one could imagine sending many thousands or even millions of bytes in a single packet. There are, however, several factors that tend to limit packet size, including: (1) the desire to limit the size of the buffers in the station for sending and receiving packets, (2) similar considerations concerning the packet buffers that are sometimes built into the Ethernet controller itself, and (3) the need to avoid tying up the channel and increasing average channel latency for other stations. Buffer management tends to be the dominating consideration. The maximum requirement for buffers in the station is usually a parameter of higher level software determined by the overall network architecture, and is typically on the order of 500 to 2000 bytes. The size of any packet buffers in the controller, on the other hand, is usually a design parameter of the controller hardware, and thus represents a more rigid limitation; to insure compatibility among buffered controllers, the Ethernet Specification mandates a maximum packet length of 1526 bytes (1500 data bytes plus overhead).

Note that the upper and lower bounds on packet length are of more than passing interest, since observed distributions are typically quite bimodal: packets tend to be either very short (control packets or packets carrying a small amount of data) or maximum length (usually some form of bulk data transfer) [Shoch and Hupp, 1979, 1980a].

The efficiency of an Ethernet system is largely dependent on the size of the packets being sent, and can be very high when large packets are used. Measurements have shown total utilization as high as 98%. A small quantum of channel capacity is lost whenever there is a collision on the wire, but the carrier sense and collision detection mechanisms combine to minimize this loss. Carrier sense reduces the *likelihood* of a collision, since the acquisition effect renders a given transmission immune to collisions once it has continued for longer than a collision interval. Collision detection limits the *duration* of a collision to a single collision interval. If packets are long compared to the collision interval, then the network is vulnerable to collisions only a small fraction of the time, and total utilization will remain high. If the average packet size is reduced, however, collision detection becomes less effective and in the limit, as the packet size approaches the collision interval, system performance degrades to that of a straight CSMA channel without collision detection. This condition only occurs under heavy load consisting predominantly of very small packets; with any "typical" mix of applications this is not a practical problem.

If the packet size is reduced still further to be *less* than the collision interval, some new problems appear. Of course, if an empty packet is already longer than the collision interval, as in the Experimental Ethernet, this case cannot arise. As the channel length, and/or the data rate are increased, however, the length (in bits) of the collision interval also increases. When it becomes

larger than an empty packet, one must decide whether stations are allowed to send *tiny* packets (smaller than the collision interval); if so, two more problems arise, one affecting the transmitter and one the receiver.

The transmitter's problem is that it can complete the entire transmission of a tiny packet before network acquisition has occurred. If the packet subsequently experiences a collision further down the channel, it is too late for the transmitter to detect the collision and promptly schedule a retransmission. In this situation, the probability of a collision has not increased, nor has any additional channel capacity been sacrificed; the problem is simply the possibility that the transmitter will occasionally fail to recognize and handle a collision. To deal with such failures, the sender of tiny packets must rely upon retransmissions invoked by a higher-level protocol, and thus suffer reduced throughput and increased delay. This occasional performance reduction is generally not a serious problem, however. Note that it is only the sender of tiny packets who encounters this behavior; there is no unusual impact on other stations sending larger packets.

The receiver's problem with tiny packets centers around its ability to recognize collision fragments by their small size and discard them. If the receiver can assume that packets smaller than the collision interval are collision fragments, it can use this to implement a simple and inexpensive fragment filter. It is important for the receiver to discard collision fragments, both to reduce the processing load at the station and to ensure that it is ready to receive the impending retransmission from the transmitter involved in the collision. The fragment filter approach is automatically valid in a network in which there are no tiny packets, such as the Experimental Ethernet. If tiny packets can occur, however, the receiver cannot reliably distinguish them from collision fragments purely on the basis of size. This means that at least the longer collision fragments must be rejected on the basis of some other error detection mechanism, such as the CRC check, or a byte or word alignment check. One disadvantage of this approach is that it increases the load on the CRC mechanism, which, while strong, is not infallible. Another problem is that the CRC error condition will now be indicating two kinds of faults: long collisions and genuine line errors. While occasional collisions should be viewed as a normal part of the CSMA/CD access procedure, line errors should not; one would therefore like to accumulate information about the two classes of events separately.

These problems, caused by tiny packets, are not insurmountable, but they do increase the attractiveness of simply legislating the problem out of existence by forbidding the sending of packets smaller than the collision interval. Thus, in a network whose collision interval is longer that an empty packet, the alternatives are:

1) *Allow tiny packets:* In this case, the transmitter will sometimes fail to detect collisions, requiring retransmission at a higher level and impacting performance. The receiver can use a partial fragment filter to discard short collision fragments (shorter than an empty packet), but longer collision fragments will make it through this filter, and must be rejected on the basis of other error checks, such as the CRC check, with the resultant jumbling of the error statistics.

2) *Forbid tiny packets:* In this case, the transmitter can always detect a collision and perform prompt retransmission. The receiver can use a fragment filter to automatically discard all packets shorter than the collision interval. The disadvantage is the imposition of a minimum packet size.

Unlike the Experimental Ethernet, the Ethernet Specification defines a collision interval longer than an empty packet, and must therefore choose between these alternatives; the choice is to forbid tiny packets by requiring a minimum data field size of 46 bytes. Since we expect that Ethernet packets will typically contain internetwork packet headers and other overhead, this is not viewed as a significant disadvantage.

6. Controller-to-Station Interface Design

We now describe a number of subtle issues concerned with designing the controller-to-station interface. The properties of this interface can dramatically affect the reliability and efficiency of systems based on Ethernet.

Turning the controller on and off. A well-designed controller must be able to (1) keep the receiver on in order to catch *back-to-back packets* (those separated by some minimum packet spacing), and (2) receive packets a station transmits to itself. We now look in detail at these requirements, and techniques for satisfying them.

Keeping the receiver on. The most frequent cause of a lost packet has nothing to do with collisions or bad CRCs. Packets are usually missed simply because the receiver was not listening. The Ethernet is an asynchronous device that can present a packet at any time, and it is important that higher-level software keep the receiver enabled.

The problem is even more subtle than that, for even when operating normally there may be periods in which the receiver is not listening. There may be *turnaround times* between certain operations when the receiver is left turned off. For example, a *receive-to-receive turnaround* takes place after one packet is received, and before the receiver is again enabled. If the design of the interface, controller, or station software keeps the receiver off for too long, arriving packets may be lost during this turnaround. This occurs most frequently in servers on a network, which may be receiving packets from several sources in rapid succession. If *back-to-back* packets come down the wire, the second one will be lost in the receive-to-receive turnaround time. The same problem can occur within a normal workstation, for example, if a desired packet immediately follows a broadcast packet—the workstation gets the broadcast, but misses the packet specifically addressed to it. Higher-level protocol software will presumably recover from these situations, but the performance penalty may be severe.

Similarly, there may be a *transmit-to-receive turnaround time* when the receiver is deaf. This is determined by how long it takes to get the receiver back on after sending a packet. If, for example, a workstation with a slow transmit-to-receive turnaround sends a packet to a well-tuned server, the answer may come back before the receiver is enabled again. No amount of retransmission by higher-levels will ever solve this problem!

It is important to minimize the length of any turnaround times when the receiver might be off. There also may be receive-to-transmit and transmit-to-transmit turnaround times; their impact on performance is not as critical.

Sending to itself. A good diagnostic tool for a network interface is the ability of a station to send packets to itself. The further the packets can get out into the network, the more the system will get tested. One of the best ways to do that is for a station to send a packet addressed to itself, and verify that the network can deliver it.

The Ethernet channel is, in some sense, half-duplex—there is normally only one station transmitting at a time. There is a temptation, therefore, to also make the controller half-duplex—that is, unable to send and receive at the same time. If possible, however, the design of the interface, controller, and host software should allow a station to send packets to itself.

Recommendations. The Ethernet Specification includes one specific requirement that helps to solve the first of these problems: there must be a minimal inter-packet spacing on the cable of 9.6 microseconds. This requirement applies to a transmitter getting ready to send a packet, and does not necessarily mean that all receivers conforming to the Specification must receive two adjacent packets. This requirement at least makes it *possible* to build a controller which can receive adjacent packets on the cable.

Satisfying the two requirements described earlier involves use of two related features in the design of a controller: full-duplex interfaces, and back-to-back receivers. A full-duplex interface allows the receiver and the transmitter to be started independently. A back-to-back receiver has facilities to automatically restart the receiver upon completion of a reception. Limited back-to-back reception can be done with two buffers: the first catches a packet, and then the second catches the next without requiring the receiver to wait. Generalized back-to-back reception can be accomplished with the use of chained I/O commands: the receiver is driven by a list of free input buffers, taking one when needed. These two notions can be combined to build any of the following four interfaces: (1) half-duplex interface, (2) full-duplex interface, (3) half-duplex interface with back-to-back receive, (4) full-duplex interface with back-to-back receive.

The Experimental Ethernet controller for the Alto is half-duplex, and runs only in a transmit or receive mode, and must be explicitly started in each mode. The need to explicitly start the receiver (that is, there is no automatic hardware turnaround) means that there may be lengthy turnaround times in which packets may be missed. This approach allows sharing certain components, like the CRC function and the FIFO.

Experimental Ethernet controllers built for the PDP-11 and the Nova are full-duplex interfaces. The transmit-to-receive turnaround has been minimized, but there is no provision for back-to-back packets.

The Ethernet controller for the Xerox 8000 processor is a half-duplex interface with back-to-back receive. Although it cannot send to itself, the transmit-to-receive turnaround delay has been avoided by having the hardware automatically revert to the receive state when a transmission is completed.

The Experimental Ethernet and Ethernet Specification controllers for the Dolphin are full-duplex interfaces with back-to-back receivers. They are the ultimate in interface organization.

Our experience shows that any one of the four alternatives will work. However, we strongly recommend that all interface and controller designs support full-duplex operation, and provide for reception of back-to-back packets (chained I/O).

Buffering. Depending upon the particular data rate of the channel and the characteristics of the station, the controller may have to provide suitable buffering of packets. If the station can keep up with the data rate of the channel, only a small FIFO may be needed to deal with station latency. If the station cannot sustain the channel data rate it may be necessary to include a full packet buffer as part of the controller. For this reason, full compatibility across different stations may necessitate the specification of a default maximum packet length. Stations can use a higher-level protocol to negotiate maximum lengths in excess of the default: if both stations and all intervening internetwork routers (gateways) can support larger packets, they can then proceed on that basis.

If a single packet buffer is provided in the controller (a buffer which has no marker mechanism to distinguish boundaries between packets), it will generally be impossible to catch back-to-back packets, and in such cases it is preferable to have at least two input buffers.

Packets in memory. The controller-to-station interface defines the manner in which data received from the cable is stored in memory, and conversely, how data stored in memory is transmitted on the cable. There are many ways in which this "parallel-to-serial" transformation may be defined [Cohen, 1981]. The Ethernet Specification defines a packet on the cable to be a sequence of 8-bit bytes, with the least significant bit of each byte transmitted first. Higher-level protocols will in most cases, however, define data types that are multiples of 8-bits. The parallel-to-serial transformation will be influenced by the programming conventions of the station and by the higher-level protocols. Stations with different parallel-to-serial transformations that use the same higher-level protocol must make sure that all data types are viewed consistently.

Type field. An Ethernet packet can encapsulate many kinds of client-defined packets. Thus, the packet format only includes a data field, two addresses, and a *type field.* The type field identifies the specific client-level protocol which will interpret the data encapsulated within the packet. The type field is never processed by the Ethernet system itself, but can be thought of as an escape, providing a consistent way to specify the interpretation of the rest of the packet.

Low-level system services, such as diagnostics, bootstrapping, loading, or specialized network management functions can take advantage of the identification provided by this field. In fact, it is possible to use the type field to identify all the different packets in a protocol architecture. In general, however, we recommend that the Ethernet packet encapsulate higher-level internetwork packets. Internetwork router stations, might concurrently support a number of different internetwork protocols, and the use of the type field allows the internetwork router to encapsulate different kinds of internetwork packets for local network transmission [Shoch, *et al,* 1980]. The use of a type field in the Ethernet packet is an instance of a principle we apply to all layers in a protocol architecture—a type field is used at each level of the hierarchy to identify the protocol used at the next higher level; it is the bridge between adjacent levels. This results in an architecture that defines a layered tree of protocols.

The Experimental Ethernet design uses a 16-bit type field. This has proven to be a very useful feature, and has been carried over into the Ethernet Specification.

7. Summary and Conclusions

We have highlighted a number of important considerations that affect the design of an Ethernet local computer network, and have traced the evolution of the system, from a research prototype to a multi-company standard, by discussing strategies and tradeoffs between alternative implementations.

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. Experience with the Experimental Ethernet in building distributed systems that support electronic mail, distributed filing, calendar systems, and other applications have confirmed many of our design goals and decisions [Birrell, *et al.*, 1981; Sturgis, *et al.*, 1980; Gifford, 1979; Shoch and Hupp, 1980b].

Four broad architectural considerations that have influenced the functions provided by the Ethernet are: reliability, addressing, priority, and encryption. We briefly touch upon them here; it is important to note that some functions are better left out of the Ethernet itself, and implemented at higher levels in the architecture.

All systems should be reliable, and network-based systems are no exception. We believe that reliability must be addressed at each level in the protocol hierarchy: each level should provide only what it can guarantee at a reasonable price. Our model for internetworking is one in which reliability and sequencing are performed using end-to-end transport protocols. Thus, the Ethernet provides a *best effort, datagram* service. The Ethernet has been designed to have very good error characteristics, and, without promising to deliver all packets it will deliver a very large percentage of offered packets without error; it includes error detection procedures, but provides no error correction.

We expect internetworks to be very large; many of the problems in managing them can be simplified by using absolute station addresses that are directly supported within the local network. Thus, address fields in the Ethernet Specification seem to be very generous—well beyond the number of hosts that might connect to one local network, but meant to efficiently support large internetwork systems.

Our experience indicates, that for practically all applications which fall into the category "loosely-coupled distributed system," the average utilization of the communications network is low. The Ethernet has been designed to have excess bandwidth; not all of which *must* be utilized. Systems should be engineered to run with a sustained load of no more than 50%. As a consequence the network will generally provide high throughput of data with low delay, and there are no priority levels associated with particular packets. Designers of individual devices, network servers, and higher-level protocols are free to develop priority schemes for accessing particular resources.

Protection, security, and access control are all system-wide functions for which there must be a comprehensive strategy. The Ethernet system itself is not designed to provide encryption, or other mechanisms for security, since by itself these techniques do not provide the kinds of protection most users require. Security in the form of encryption, where required, is the responsibility of the end-

user processes.

Higher-level protocols raise their own issues of compatibility over and above those addressed by the Ethernet and other link-level facilities. 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. We expect to see standards for higher-level protocols emerge during the next few years.

Within an overall distributed systems architecture, the two generations of Ethernet systems have proven to be very effective local computer networks.

8. Acknowledgements

Many people have contributed to the success and evolution of the Ethernet local computer network. Bob Metcalfe and David Boggs built the Experimental Ethernet at the Xerox Palo Alto Research Center, and Tat Lam built and supplied the many transceivers. Since then Ed Taft, Hal Murray, Will Crowther, Roy Ogus, Bob Garner, Ed Markowski, Bob Printis, Bob Belleville and Bill Gunning have contributed to the design and implementation of the Ethernet. The joint cooperation between Digital Equipment Corporation, Intel Corporation, and Xerox Corporation also produced many important contributions to the Ethernet Specification.

9. References

[Birrell, *et al.*, 1981]
A. D. Birrell, R. Levin, R. M. Needham and M. D. Schroeder, "Grapevine," to be presented at the *Eighth Symposium on Operating Systems Principles*, 1981.

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

[Boggs and Metcalfe, 1978]
D. R. Boggs and R. M. Metcalfe, *Communications network repeater*, United States Patent No. 4,099,024, July 4, 1978.

[Boggs, *et al.*, 1980]
D. R. Boggs, J. F. Shoch, E. A. Taft, and R. M. Metcalfe, "PUP: An internetwork architecture," *IEEE Transactions on Communications*, com-28:4, April 1980, pp. 612-624.

[Cerf and Kirstein, 1978]
V. G. Cerf and P. K. Kirstein, "Issues in Packet-Network Interconnection," *Proceedings of the IEEE*, vol 66, no 11, November 1978, pp. 1386-1408.

[Cohen, 1981]
D. Cohen, "On Holy Wars and a Plea for Peace," to be published in *IEEE Computer* magazine, 1981.

[Crane and Taft, 1980]
R. C. Crane and E. A. Taft, "Practical considerations in Ethernet local network design," *Proc. of the 13th Hawaii International Conference on Systems Sciences*, January 1980, pp. 166-174.

[Dalal, 1981]
Y. K. Dalal, "The Information Outlet: A new tool for office organization," *Proceedings of the Online Conference on Local Networks and Distributed Office Systems*, London, May 1981, pp. 11-19.

[Dalal and Printis, 1981]
Y. K. Dalal and R. S. Printis, "48-bit Internet and Ethernet Host Numbers," to be published in the *Proceedings of the Seventh Data Communications Symposium*, October 1981.

[Dorado, 1981]
The Dorado: A High-Performance Personal Computer. Three Reports, Xerox Palo Alto Research Center, CSL-81-1, January, 1981.

[Ethernet, 1980]
The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications, Version 1.0, September 30, 1980.

[Ethernet, 1981]
The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications, Version 2.0, to be published in 1981.

[Gifford, 1979]
D. K. Gifford, "Violet, an Experimental Decentralized System," Xerox Palo Alto Research Center, CSL-79-12, September, 1979.

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

[Metcalfe, 1973a]
R. M. Metcalfe, "Steady-State Analysis of a Slotted and Controlled Aloha System with Blocking," *Proceedings of the Sixth Hawaii Conference on System Sciences*, January 1973. Reprinted in the *SIGCOM Review*, January 1975.

[Metcalfe, 1973b]
R. M. Metcalfe, *Packet Communication*, Ph.D. Thesis Harvard University, Project MAC Report MAC TR-114, MIT, December 1973.

[Metcalfe and Boggs, 1976]
R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Communications of the ACM*, 19:7, July 1976, pp. 395-404.

[Metcalfe, *et al.*, 1977]
R. M. Metcalfe, D. R. Boggs, C. P. Thacker, and B. W. Lampson, *Multipoint data communication system with collision detection*, United States Patent No. 4,063,220, December 13, 1977.

[Rawson and Metcalfe, 1978]
E. G. Rawson and R. M. Metcalfe, "Fibernet: Multimode Optical Fibers for Local Computer Networks," *IEEE Transactions on Communications*, com-26:7, July 1978, pp. 983-990.

[Schoute, 1977]
F. C. Schoute, *Decentralized Control in Computer Communication*, Technical Report No. 667, Division of Engineering and Applied Physics, Harvard University, April 1977.

[Shoch, 1978]
J. F. Shoch, "Internetwork Naming, Addressing, and Routing," *17th IEEE Computer Society International Conference (Compcon)*, September 1978, pp. 430-437.

[Shoch, 1980]
J. F. Shoch, *An annotated bibliography on local computer networks (3rd edition)*, Xerox Parc Technical Report SSL-80-2, and IFIP Working Group 6.4 Working Paper 80-12, April 1980.

[Shoch, in press]
J. F. Shoch, *Local Computer Networks*, McGraw-Hill, in press.

[Shoch and Hupp, 1979]
J. F. Shoch and J. A. Hupp, "Performance of an Ethernet local network—a preliminary report," *Local Area Communications Network Symposium*, Boston, May 1979, pp. 113-125. Revised version presented at the *20th IEEE Computer Society International Conference (Compcon '80 Spring)*, San Francisco, February 1980, pp. 318-322.

[Shoch and Hupp, 1980a]
J. F. Shoch and J. A. Hupp, "Measured performance of an Ethernet local network," *Communications of the ACM*, 23:12, December 1980, pp. 711-721.

[Shoch and Hupp, 1980b]
J. F. Shoch and J. A. Hupp, "Notes on the "Worm" programs—some early experiences with a distributed computation," Xerox Palo Alto Research Center, SSL-80-3, September, 1980.

[Shoch, *et al*, 1980]
J. F. Shoch, D. Cohen, and E. A. Taft, "Mutual Encapsulation of Internetwork Protocols," *Proceedings, Trends and Applications: 1980—Computer Network Protocols*, May 1980, pp. 1-11. Revised version to appear in *Computer Networks.*

[Sturgis, *et al.*, 1980]
H. Sturgis, J. Mitchell, and J. Israel, "Issues in the Design and Use of a Distributed File System," *ACM Operating Systems Review*, 14:3, July 1980, pp. 55-69.

[Thacker, *et al.*, 1979]
C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs, *Alto: A personal computer*, Xerox Palo Alto Research Center Technical Report CSL-79-11, August 1979. To appear in Siewiorek, Bell, and Newell (Eds.), *Computer Structures: Readings and Examples, 2nd edition.*

[Zimmermann, 1980]
H. Zimmermann, "OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection, *IEEE Transactions on Communications*, com-28:4, April 1980, pp. 425-432.

OPD-T8102

XEROX

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

 Printed in U.S.A.