Practical Considerations in Ethernet Local Network Design

by Ronald C. Crane and Edward A. Taft

October 1979, revised February 1980

Abstract: The design of a useful local-area communications network requires careful attention to a number of practical issues in addition to the theoretical concerns and analytical models emphasized in existing literature.

This report considers the design of the Ethernet local network, a broadcast multi-access bus system using carrier sense and collision detection. The design must be measured against the goals of the overall communications architecture which it supports, so the important properties of this architecture are briefly presented. The Ethernet design is then described in some detail, with emphasis on construction of the channel, interfaces, and low-level protocols. The report concludes with a summary of experience with an operational system and some suggestions for further improvement.

CR Categories: 3.81, 6.35

Key words and phrases: Local-area network, carrier-sense multi-access with collision detection (CSMA/CD), communications interface

A condensed version of this paper was presented at the Hawaii International Conference on System Sciences, *January 1980.*

XEROX
SYSTEMS DEVELOPMENT DIVISON and
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

1. Introduction

This report considers the design and implementation of the "Ethernet" local-area communications network. This is a passive bus-oriented system, with distributed control based on carrier-sense multi-access with collision detection (CSMA/CD). First described in [Metcalfe & Boggs, 1976], this architecture has been the subject of considerable analysis and has inspired numerous variations.

Very little, however, has been written on the actual implementation of Ethernet-style local networks. Past emphasis has been on the theory of operation and on modelling and analyzing the communications channel itself. But in a practical system, the choices made in the construction of the channel, interfaces, and low-level protocols are of crucial importance.

In this report, we begin in section 2 by presenting briefly the overall communications architecture that the Ethernet system is intended to support. Section 3 describes in some detail the prototype Ethernet implementation and its interface to a particular computer, and highlights some areas in which special care is required. Section 4 presents some operational considerations that are consequences of the present design. And in the final section we summarize our experience with an operational Ethernet system, discuss the strengths and weaknesses of the present design, and point out some areas requiring further work.

2. Overall design principles

The design of a local computer network must be considered in the context of the overall communications architecture it is intended to support. It is too frequently the case that communications systems are designed bottom-up and piecemeal, with results that are needlessly complex or are a poor match for the intended applications [McQuillan, 1979].

In the case of the Ethernet system, the context is an *internetwork* architecture called "Pup" [Boggs *et al*, 1980]. As in other such designs [Cerf & Kahn, 1974; Sunshine, 1977], it is the purpose of the Pup internet to facilitate computer-to-computer communication among diverse host systems interconnected by a heterogeneous collection of networks. These networks are coupled by *internetwork gateways*, which are hosts that are connected to more than one network and are willing to forward packets among them.

2.1 The computational and communications environment

The computing environment addressed by the Pup architecture consists primarily of "Alto" minicomputers [Thacker *et al.*, 1979] and other personal computers capable of high-quality interaction with human users in many application areas—text editing, illustration, document preparation, interpersonal communication, and others. These machines appear in clusters, sometimes with over 100 Altos in one *campus* (building or group of adjoining buildings). Each

campus is supported by various specialized servers providing remote access to shared resources such as large file systems, magnetic tapes, and high-quality printers. Efficient, high-bandwidth communication within a campus is essential.

The campuses are interconnected by slower communication facilities, consisting mostly of leased telephone circuits, but also including other networks such as the Arpanet and the ARPA Packet Radio network. These facilities must also support the full generality of internetwork communication, just with lower performance.

2.2 Pup principles

This section presents the highlights of the Pup internetwork architecture, in preparation for examining their impact on local network design. These topics are elaborated upon in [Boggs *et al.*, 1980].

Simplicity. A guiding principle of the Pup design has been the desire for simplicity, especially at the lower levels. The lowest level of internet protocol is a simple end-to-end datagram—a standardized packet (called a "Pup") which is media-independent but otherwise retains the common properties of the underlying packet networks. Maintaining simplicity in the low-level design facilitates introduction of new network technology and best accomodates new applications. Furthermore, the computational environment is dominated by relatively lightweight hosts requiring high-bandwidth intercommunication, so simplicity of the basic protocols is essential to efficient implementations.

End-to-end protocols. The basic facility provided by the Pup internet is the transport of independently addressed Pups from a source process to a destination process. All higher-level interprocess communication primitives, such as connections, streams, transactions, etc., are implemented by higher-level protocols that are strictly a matter of agreement among the communicating end processes. This arrangement makes the internet itself very simple, imposes minimal functional requirements on the underlying packet networks, and contributes to reliability by keeping fragile end-to-end state out of intermediate agents such as gateways.

Individual networks viewed as packet transport mechanisms. The only capability required of an underlying packet network is the ability to transport Pups from a source to a destination within that network. To this end, an *encapsulation* technique is defined for each type of network, permitting a Pup to be carried transparently. Depending on the requirements of the network, encapsulation may include various transformations such as adding headers and trailers, performing fragmentation, applying error correction or other coding techniques, and so on. The Pup specification has nothing to say about such transformations except that they be *inverted* as an encapsulated Pup leaves the network.

Reliability of transport. Transporting networks are *not* required to be perfect; rather, they should simply give their "best efforts" to deliver each Pup [Metcalfe, 1973]. Packets may be lost, duplicated, delivered out of order, after a great delay, and with hidden damage. For performance reasons, the underlying networks should be designed to make such events uncommon. But it is the responsibility of the end processes to attain the level of reliability *they* require, by use of suitable end-to-end protocols.

2.3 Consequences for local network design

In light of the principles of the Pup architecture, it is clear that the functional capability required of a local network is really quite simple: the transport of an encapsulated, individually addressed Pup between any two hosts connected to the network, with good but not necessarily perfect reliability. Other capabilities that might be engineered into a local network are of little use in the Pup design because they are generally difficult to extend into the full internet environment.

Given this relatively simple functional requirement, one can concentrate on other important aspects of a local network design. These include *availability*, the network's resistance to catastrophic failure (as opposed to isolated packet loss, which is of little consequence); *performance*, the extent to which the network's capacity is accessible to the end communicating processes; and *maintainability*, the cost and effort required for installation and maintenance.

These properties of a local network are of course heavily dependent on the theory that underlies the local network design. In the case of the Ethernet system, that basis is carrier-sense multi-access with collision detection (CSMA/CD). That technique has by now received fairly extensive theoretical analysis [Metcalfe & Boggs, 1976; Agrawala *et al.*, 1977; Almes & Lazowska, 1979; Tobagi & Hunt, 1979].

Equally important are a number of considerations that arise in the design and engineering of a functioning and useful local network. These considerations apply not only to the communications channel itself but also to the transceivers, controllers, and low-level software and protocols, as discussed in the next section.

3. Ethernet design and implementation

We now describe the existing Ethernet system: the shared channel, the interface to the Alto minicomputer (the predominant type of Ethernet host), and the low-level protocols. We start with a brief summary of system operation and then proceed with the component descriptions.

3.1 Theory of operation

In a CSMA/CD system, stations *contend* for use of the "Ether" (common broadcast communications channel) until one of them *acquires* and uses it to transmit a packet. *Carrier sense* is used to *defer* transmission until the Ether is quiet (no other carriers present). When quiet is detected, the source machine proceeds to transmit.

During transmission, the source listens for a *collision* (any other carrier on the Ether at the same time). Since other stations will defer once they see a carrier, the first round-trip propagation time is the only time that collisions can occur in the network. If no other carrier is detected within one-round trip time, the source is said to have *acquired* the Ether. If another carrier is detected, the packet is aborted and a *jam* signal is transmitted to ensure that the collision is also detected by all other participants in it. After jamming, each station waits a random length of time and tries again. Justification for these design choices and much more detail may be found in [Metcalfe & Boggs, 1976; Shoch, 1979].

3.2 Ethernet system overview

The prototype Ethernet system is a bus-oriented communications network providing a raw bandwidth of 3 megabits per second, connecting up to 255 computers, and spanning a linear distance of 1 kilometer. A piece of 75-ohm coaxial cable with terminators on each end constitutes the bus. The cable is typically located in the ceiling and travels over or near areas to be served by the network. The overall structure of this system is illustrated in figures 1 and 2.

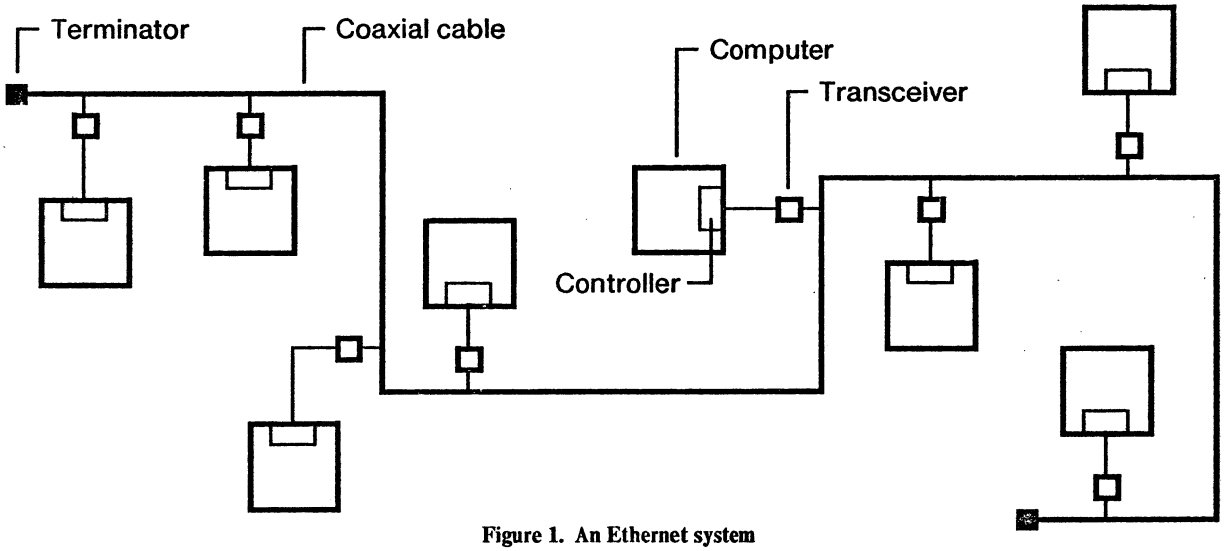

Figure 1. An Ethernet system

Transceivers are located next to the cable and are attached to it by pressure taps. The transceiver provides ground isolation and a high-impedance connection to the cable for transmitting and receiving. Twisted pair cable brings signals and power to the transceiver from a controller in

the computer. This portion of the Ethernet system is identical for all types of hosts.

The Ethernet controller for the Alto computer is implemented partially in hardware, partially in microcode, and partially in software. The hardware contains a small (16-word) buffer, parallel-serial and serial-parallel converters, CRC generator and checker, and Manchester phase encoder and decoder. The microcode performs address recognition, schedules retransmissions when collisions occur, and provides the interface seen by the software. The software is responsible for basic packet formatting, Pup encapsulation and decapsulation, and interfacing to the network-independent software, which in turn distributes and collects Pups (internetwork datagrams) to and from the end communicating processes.

Following is an elaboration of specific details of the system implementation.

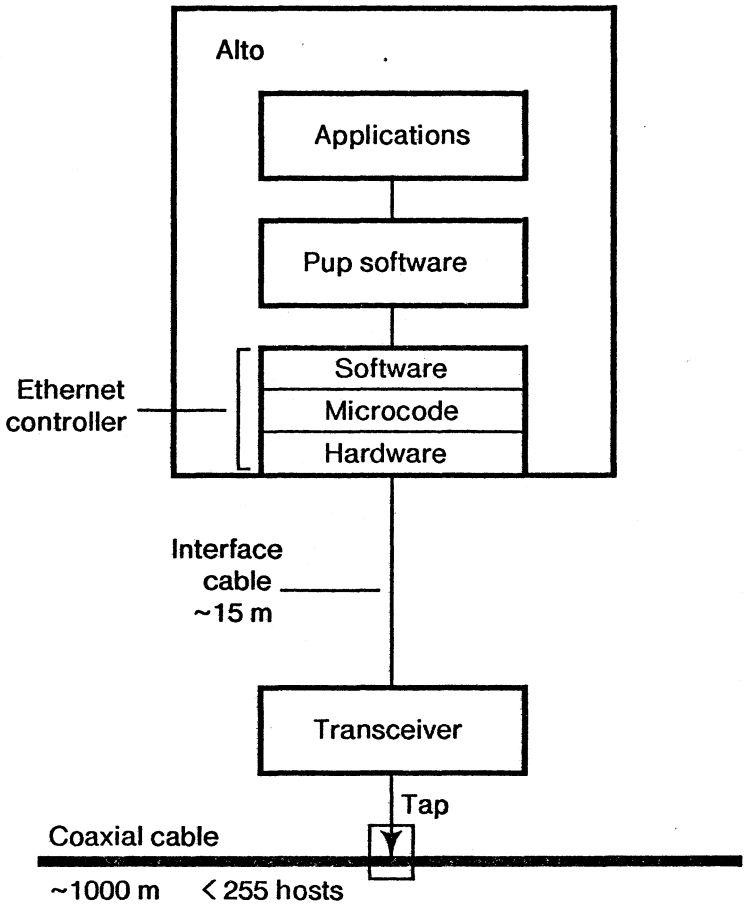

Figure 2. Alto-Ethernet connection

3.3 Cable and taps

Cable selection has been based on low signal loss and the ability to be tapped using existing cable TV technology. A foam type RG/11 coaxial cable is used for the Ether. Transceivers are connected to it using a cable tap manufactured for the cable television industry by Jerrold

Electronics. It comes in two parts: a split metal block with two pins which clamps the cable and pierces the shield, and a 3/8-inch diameter coaxial pin assembly (tap) which screws into a threaded hole in the block, pierces the core dielectric of the cable, and contacts the center conductor.

The common coaxial cable and the individual transceiver-to-controller cables are typically located in the space above the false ceilings in office buildings. While these cable systems utilize less than 30 volts and are current-limited, thereby exempting them from many building regulations, use of the space above hung ceilings as an air return duct (plenum) for the building ventilation system exposes the system to stringent fire regulations.

The critical parameters for approval of the cable are its ability to propagate a flame and the number of pounds of combustible material that it contains. Teflon requires a 95 percent oxygen atmosphere to burn, very high temperatures to cause decomposition, and has a relatively low energy content per pound. Teflon coaxial cable has recently received UL approval for use in air plenums [Du Pont, 1979].

3.4 Transceiver

The transceiver contains a small quantity of electronics to drive and receive signals on the cable, detect collisions, maintain ground isolation, and protect the shared channel from certain failures of the transceiver, controller, or host. Figure 3 illustrates the functions of the transceiver.

Signalling Conventions. Data on the cable is Manchester encoded. This coding has the property that it has a transition in every bit cell and has a 50 percent duty cycle. Bits are phase-encoded in the controller before being passed to the transceiver. The first half of a bit cell contains the complement of the bit and second half contains the bit itself. There is always a transition in the middle of the bit cell: a positive edge corresponds to a "one" bit and a negative edge to a "zero". The voltage levels transmitted into the cable are +3 volts for "on" and 0 volts for "off". Carrier is detected by the presence of transitions on the cable; when no transmitter is active, the channel is quiescent in the "off" state.

Collision detection. The transceiver contains a circuit that performs the function of collision detection. It compares the received data with that being transmitted and (after suitable filtering to eliminate spikes due to skew) produces a *collision* signal whenever there is a difference. (The host's controller monitors *collision* only while it is transmitting.) The circuit is located in the transceiver rather than the controller because that is the point at which the skew between transmitted and received signals is at a minimum.

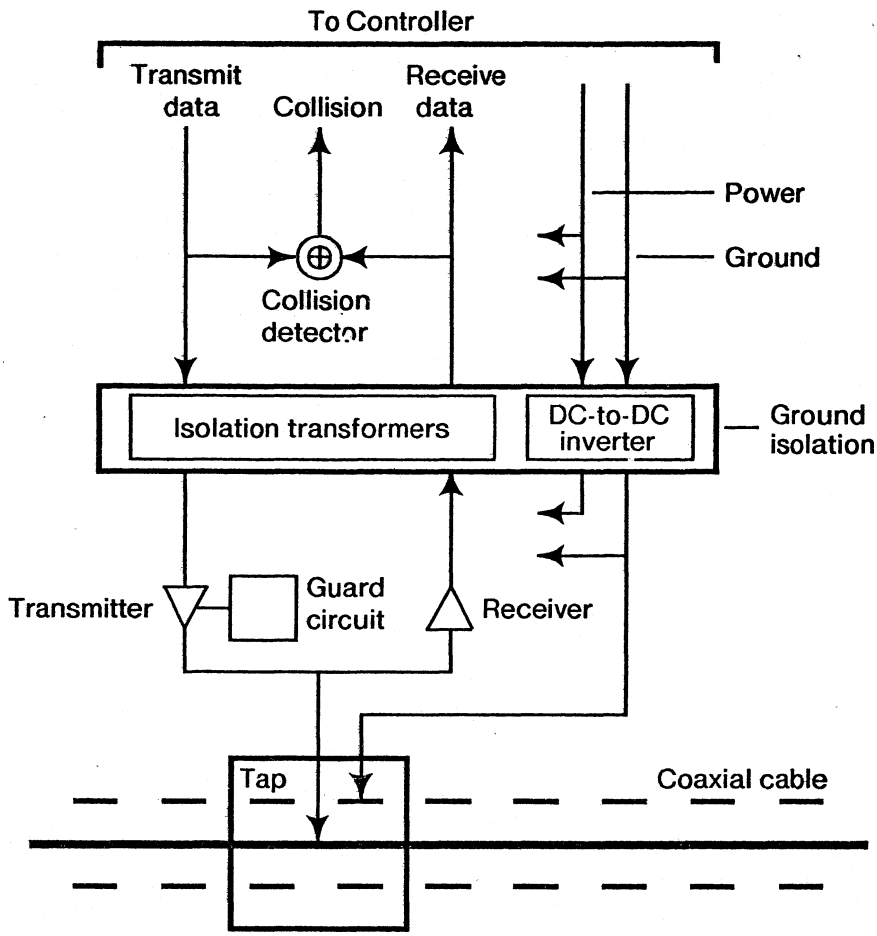

Figure 3. Ethernet Transceiver

This method of detecting collisions depends on the fact that with Manchester coding, the transmitter is off for half of each bit period. If some other host is transmitting simultaneously, any of its "on" intervals that overlap this transmitter's "off" intervals will be detected as collisions. An exclusive-OR gate is used in the Ethernet transceiver. This technique is limited to receivers using a fixed DC threshold for a reference. Under conditions of greater attenuation, higher data rate, or longer cables, an analog hybrid would be required to generate the collision signal.

Ground isolation. There are two ground references in the system. One is the common coaxial cable shield and the other is the local ground associated with each Alto or other host computer. It is important that these not be tied together, since one local ground may differ by several volts (60 Hz) from another local ground. Connection of several local grounds to the common cable could cause a large current to flow through the cable's shield. The ground isolation provided by the transceiver eliminates this connection through an Alto. It is equally important to prevent this connection via contact of a tap block with building grounds such as cable trays, conduits, or ceiling hangers. (Later we describe one consequence of failure to observe this isolation.)

Protection against failures. Each transceiver provides a high-impedance connection to the cable in both the power-on and power-off states. It contains a guard against transceiver circuit failure, which monitors the length of transmitted pulses and latches the unit off if they are too long. (This may be reset by removing power for several seconds.) Additionally, the AC coupling between the host computer and the transceiver prevents the channel from being permanently jammed by a controller whose *transmit data* signal becomes stuck on.

The transceiver electronics themselves are subject to damage from electrical transients in the cable. In more recent designs, transceivers have been modified to withstand 20 to 30 volts between the center conductor and shield.

RF emissions. Consideration of RF emissions is not only a courtesy to other electronic equipment and radios but is also necessary to meet FCC regulations. RF emissions may arise from the interface cable between controller and transciever, from the transceiver itself, and from the tap block connection. All of these sources can be reduced to acceptable levels.

The existing system uses unshielded twisted-pair between the transceiver and the controller, driven in a single-ended manner. A significant improvement results with use of balanced differential signalling and an overall shield.

Improper design of the transceiver isolation stage can cause a common-mode voltage to appear between its connection to the interface cable ground and its connection to the shield of the coaxial cable. The introduced signal may be at either the signal frequency or the switching frequency of the DC-to-DC inverter or both, depending on the design. These issues must be addressed at design time, as they are very difficult to correct after the unit is built.

It is important to minimize the impedance between the tap block ground and the cable shield, because a small voltage develops across this impedance whenever a transceiver is transmitting.

Signal symmetry. It is essential that the symmetry of the signal be preserved as it travels from a controller into a transceiver, through the coaxial cable, back into a transceiver, and into a receiving controller. Considerable care has been taken in the design of the cable drivers and receivers for both the interface cable and the common coaxial cable.

On the coaxial cable, transmit pulses are 3 volts in amplitude on the cable and coupled into the cable through a diode. The receiver is DC-coupled to the cable and has threshold of about 1.3 volts, which is the centerpoint of a waveform received 1 kilometer from the transmitter. Near the transmitter the threshold should be 1.5 volts, but there the rise and fall times of the signal are only 12 nanoseconds, so the threshold voltage is not as important nearby as it is at the end of the cable where the signal waveform looks more like a sine wave.

The second part of maintaining symmetry is the method of driving and terminating the interface cable. The present system uses open-collector drivers and a receiving end composed of a TTL gate and a termination impedance equal to the cable impedance and tied to a voltage twice the threshold voltage of the TTL gate (approximately 2.8 volts). This provides a symmetrical signal about the threshold and prevents reflections in the cable.

3.5 Controller

While controllers have been implemented for several computers, the Alto controller, whose hardware is diagrammed in figure 4, is the most widespread at present.

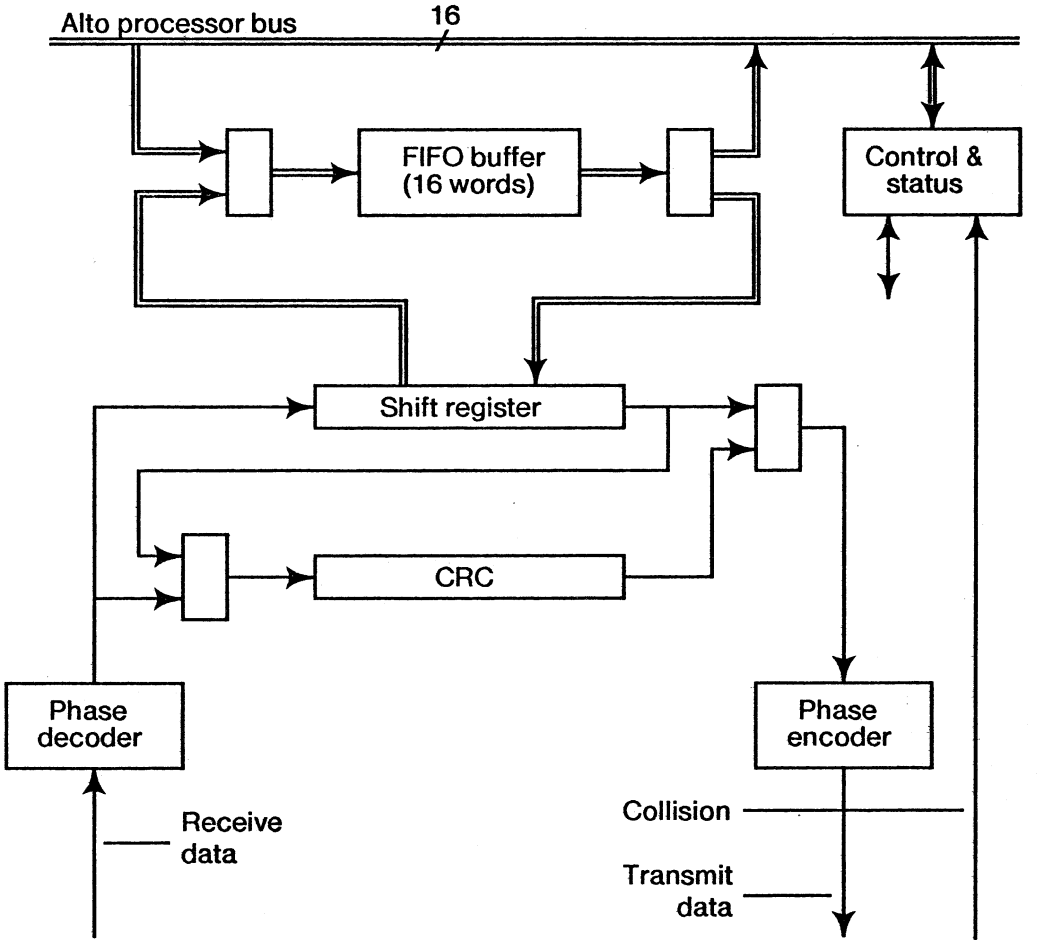

Figure 4. Alto-Ethernet controller hardware. Single lines are serial data paths. Double lines are 16-bit parallel data paths. This controller is half-duplex and is configured either to transmit or to receive at any given time. Newer designs have independent transmit and receive paths.

Controller hardware considerations. The size and complexity of a controller is dependent on the architecture and speed of the machine to which it is interfaced. The I/O architecture determines the amount of hardware needed to make a connection to the machine, and the machine's speed determines whether some of the functions could be performed by microcode or software instead of hardwired logic. In the case of the Alto, the hardware is quite simple because the Alto's microprogrammable processor controls I/O devices directly, and data transfers, address filtering, and exception handling are performed by microcode. (For an elaboration on this, see [Thacker *et al.*, 1979].)

The controller must provide adequate internal buffering of bits to accomodate the bandwidth and latency of the host computer's I/O system. Since high-speed buffer chips are expensive, the smallest amount of buffering that meets this requirement is used. In all of the systems we have built so far, the I/O bandwidth of the processor exceeds the bit rate of the channel and has a service latency time considerably less than a packet transmission time. Thus, buffers may be small—16 words in the Alto. In an interface to a processor which has a low I/O bandwidth, the buffer would have to be able to hold an entire packet, and additionally the packet rate to that machine would have to be sufficiently slow to allow the machine to empty the buffer before the next packet arrived.

It is possible for the host to lose incoming packets if the controller fails to listen to the Ether all the time. This is most prevalent in half-duplex controllers (such as the one used in the Alto) where there is significant turnaround time between ending a local transmission and restarting the receiver. The maximum turnaround delay without packet loss determines the minimum allowable packet separation on the cable.

In the prototype Ethernet system, minimum interpacket separation is only a few bit times. In the Alto controller, there is a microcode/software turnaround delay considerably larger than this; consequently, the Alto does miss some incoming packets, with performance ramifications that will be discussed later (section 5.1). This is discussed at greater length in [Shoch, 1979].

A design in which the hardware automatically switches modes at the completion of transmission eliminates this problem, provided that the microcode and software are prepared to deal with an incoming packet right on the heels of one just transmitted. A full-duplex interface, with independent transmit and receive data paths, is structurally simpler (though requiring slightly more hardware); it has the added advantage of enabling the host to receive its own transmissions, which is useful for self-testing and diagnosis. Most of our newer designs are full-duplex.

Phase decoding techniques. Each controller contains a circuit that recovers data and clock from the Manchester encoded signal received from the cable. Two techniques are in use in the current Ethernet system. Both have the property that they can synchronize to the one-bit preamble of the packet and successfully receive the rest of the packet. The first type, used in the Alto, is analog in nature, using two one-shots to recover data and clock and to generate a *carrier detect* signal. The second type, used in newer machines, is digital in nature, using a finite state machine that samples the received signal at eight times the bit rate. The state machine also checks the timing between transitions, flagging malformed signals as probable collisions.

Collision handling. Here we discuss the details of contention in the network. As described previously, a transmitting station detects a collision, transmits a jam pulse, and retransmits after waiting a random interval of time. To ensure fairness, it is important that all stations use the same algorithm.

When the *collision* signal is raised by the transceiver, the controller ceases data transmission immediately and turns the transmitter continuously "on" for 3 microseconds, then shuts down. This *jam* signal ensures that all participants in the collision are made aware of it and also shut down.

When a collision occurs, a retransmission timer is started. After it expires, the controller may retransmit the packet (of course, first deferring to any ongoing transmissions in the usual manner). The timer is set to a random value computed using a *binary exponential backoff* algorithm. The value chosen is uniformly distributed between zero and $2^n - 1$, where $n = \min(retries, 8)$ and *retries* is the number of previous unsuccessful attempts to transmit the packet. The unit of time is 38 microseconds, which—in theory—is chosen to approximate the maximum contention interval or *slot* time [Metcalfe & Boggs, 1976], but in practice is simply any convenient quantum that exceeds the slot time.

The Alto implementation of the binary exponential backoff algorithm is very simple. For each transmission attempt, a mask, initialized to zero before the first attempt, is ANDed with a suitable source of 8-bit pseudo-random numbers (e.g., 8 rapidly-changing bits of an unsynchronized real-time clock) to produce the timer value. The mask is then left-shifted one bit and a "one" introduced into the low-order bit. This process repeats for each retransmission attempt until the packet is successfully transmitted. If 16 successive collisions occur, the controller abandons attempting to transmit the current packet; this *load overflow* condition is an indication of extreme overload or network failure and is left for higher-level software to deal with.

Receivers should take some care to detect and filter out packets that have suffered collisions, and to do so at a fairly low level so as not to burden higher-level software with the expense of dealing with these relatively common events. There are several ways to do this. The design parameters of the present Ethernet system (1 kilometer, 3 megabits per second) are such that collisions give rise to packets less than 32 bits long, so the controller can simply filter out such "runt packets". An additional level of redundancy is afforded by use of a 16-bit cyclical redundancy check (CRC) and by rejecting packets that do not end on a 16-bit word boundary. Newer controller designs that employ digital phase decoding can detect encoding violations, which are frequently an indication of collision.

Minimizing the probability of collision. A collision occurs when a transmitter fails to defer to another transmission already in progress. The failure is due to the time delay between start of packet transmission in one machine and the carrier sense ("decision to transmit") circuitry in the other machines. The worst case of this determines the *acquisition time* of the Ether. Most of this time is due to propagation delay through the cable, but some is due to transceiver and controller delays.

For any given packet, the acquisition time is given by the following:

$$T_a = 2\,T_p + 2\,T_d + T_s$$

where T_p is the propagation delay through the cable to the other machine; T_d is the delay through the transceiver, transceiver cable, and controller up to decision gate; and T_s is the delay from the decision to start transmission until the appearance of the start of the packet on the common cable.

The probability of a given packet colliding with any packet from some other machine transmitting n packets per second is:

$$P = n\,T_a$$

assuming $n \leq 1/T_a$ and packet transmission time $\geq T_a$. (If packet transmission time is less than T_a then the concept of acquisition breaks down, as does the fundamental concept of transmitter-based collision detection. This is the point at which Ethernet channel behavior moves toward that of a slotted Aloha channel [Abramson, 1977].)

The consequence of this with respect to controller design is that the decision to transmit should be made just before the packet would actually be sent to the cable. In particular, the timer that enforces the minimum spacing between packets should always start when an end of packet is detected (loss of carrier) instead of starting after the decision to transmit. Thus if the Ether has been quiet for a while, a transmission proceeds immediately rather than waiting for the timeout.

3.6 Low-level protocols

In the Pup architecture, the purpose of a low-level protocol is to deal with peculiarities of the network. Functions in excess of this (e.g., connections, flow control, sequencing, perfectly reliable delivery) don't belong here but rather at higher levels of protocol—which, in the Pup design, are all end-to-end. In the Ethernet system, an attempt has been made to provide the simplest possible mechanisms that have the required functionality and performance.

Packet format. The basic Ethernet packet format is shown in figure 5. A packet begins with a *sync bit*, which is always a one and whose leading edge enables the receiver to acquire bit phase. Following this are two 8-bit address fields, the *destination host* and the *source host*; these are followed by a 16-bit *packet type*, zero or more 16-bit words of *data*, and a 16-bit *cyclical redundancy check* (CRC). The sync bit and the CRC are generated and stripped off by the controller hardware; all remaining information is transferred to or from the host computer's memory.

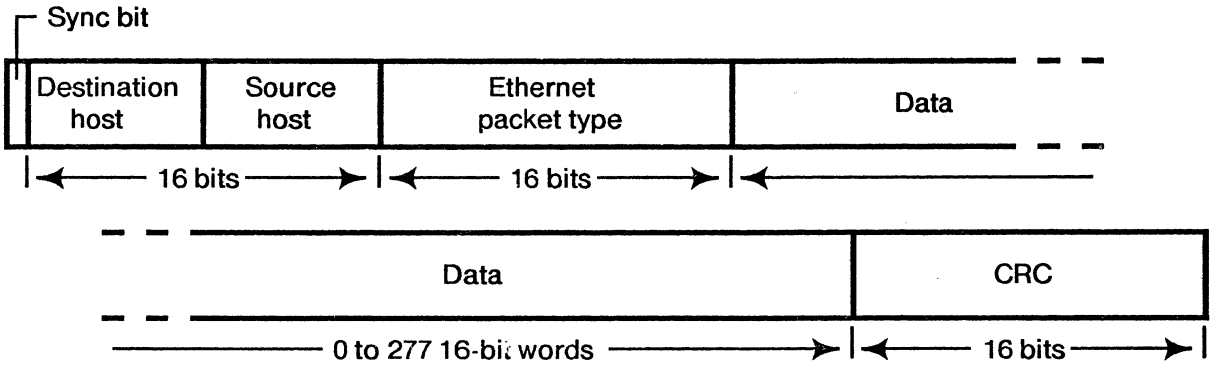

Figure 5. Ethernet packet format. Sync bit and CRC are added and removed by controller hardware. All other fields are generated and consumed by software, though the destination host is also examined by receiver microcode to perform address filtering. Maximum packet length is a software convention.

Addressing. The addressing conventions used in the prototype Ethernet system are very simple. Each host is assigned a unique 8-bit address that is readable by software running in the host, and each packet transmitted is labelled with its source and intended destination address.

Of course, the Pup internet protocol requires an address space much larger than this, extended both upward (to encompass multiple networks) and downward (to address multiple processes within a machine). Those addresses, however, are in the Pup itself—that is, encapsulated within the *data* portion of an Ethernet packet.

Received packets are copied into memory under control of an *address filter*, which in the Alto is implemented in microcode. Upon receiving the first word of a packet, the microcode compares the destination host field against an address supplied by software (normally the host address assigned to the machine), and copies the packet into memory only if they are equal.

The software can also set the address filter to be *promiscuous*, meaning that every received packet is to be copied into memory. This is useful for network monitoring, protocol debugging, and other purposes.

A destination host address of zero is used to indicate a *broadcast* packet, which every active host should copy into memory. Broadcasts are useful for locating nearby resources (on the same network) and for distributing information of general interest. But broadcasts must be used with discretion, since all hosts that receive one incur a certain amount of software overhead, if only to discard it.

A generalization of broadcast is *multicast*, in which a packet may be addressed to a predefined set of hosts, referred to as a *multicast group* and identified by a *multicast ID*. In the present Ethernet system, a form of multicast may be implemented by reserving certain addresses as multicast IDs rather than host addresses. A host then joins a multicast group by setting its address filter to accept packets addressed to the group's ID. (This simple scheme has some obvious limitations, such as the inability for a host to belong to more than one multicast group simultaneously. But the address filtering may be made arbitrarily elaborate by loading different microcode, or by setting the microcode address filter to *promiscuous* and doing address filtering in

the software.)

While broadcast and multicast are easy to implement within a single network, they are difficult to extend into the full Pup internetwork environment. Consequently, existing Ethernet systems make little use of these techniques (multicast especially), and we have intentionally avoided introducing added complexity into controllers or low-level protocols to provide for them.

Other conventions. The *packet type* field is reserved, by software convention only, to identify the specific higher-level protocol under which the *data* should be interpreted. While Pup is the predominant higher-level protocol at present (and hence most Ethernet packets identify their contents as being Pups), there do exist other protocol architectures—notably the ARPA Internet—whose basic packets may also be encapsulated within Ethernet packets. Also, there are a few applications that are highly specialized to the Ethernet or to the Alto, such as the initial step of bootstrap loading; these utilize their own Ethernet protocols rather than being based on Pup.

The Ethernet system does not employ low-level acknowledgments. Packet errors are so infrequent that low-level acknowledgments are not needed: end-to-end protocols can easily cope with the network's error rate. Likewise, no special low-level protocols are required to gain access to the network, since the channel is controlled strictly by contention.

4. Operational considerations

This section briefly highlights a few of the important operational characteristics of the present Ethernet design.

4.1 Installation

Simple rules describe the installation of the coaxial cable. It must have a terminator at each end, it must not branch, and it must never be connected to a building ground (by contact to connector casings or tap blocks). The terminators should be located in accessible locations so that a time domain reflectometer can be conveniently connected to the end for checkout. The cable should be broken every 100 meters or so and a connector installed in an accessible location; this facilitates installation of the cable and is also useful in fault isolation, as will be described.

Installation of a tap and transceiver is relatively straightforward and does not require taking the network down. The tap block is placed on the cable and tightened, piercing the jacket and making contact with the shield. The jacket and shield coring tool is screwed in and removed, leaving a hole for the tap. The transceiver with tap is screwed into the tap block, completing the job. The tap and tap block may subsequently be removed and the hole taped over with no effect on the rest of the system.

It is best to install taps at least a few feet apart so as to distribute the taps' and transceivers' capacitance and impedance along the cable rather than lumping it all in one place. In the prototype Ethernet system, running at 3 megabits per second, this is not especially important (one existing Ethernet cable, about 500 meters long and with over 120 hosts, has several clumps of closely-spaced transceivers with no ill effects); however, at higher data rates this would become a more serious concern.

4.2 System configuration

The current Ethernet design uses 8-bit host IDs assigned on a per-network basis. If a machine is moved from one network to another—a fairly common event for Altos—its ID may have to be changed to fit into the address space of the new network. This is an operational annoyance.

An alternate scheme is to provide each host with a larger ID (at least 32 bits) which is unique across all networks. This simplifies the movement of machines from one network to another. Of course, some central source of IDs must be maintained to assure that none are repeated.

Aside from assignment of host IDs, no central coordination is required when a host is added to or removed from a network.

4.3 Reliability and fault isolation

There are two classes of failures: those that disable a single host and those that take down the entire network. Because the common channel is passive, most failures of transceivers and interfaces affect only one host, and it is immediately obvious which host it is.

Network failures can be caused by a transceiver stuck in the "on" state, a shorted transceiver, or a missing terminator. All of these have the effect of reducing the signal to noise ratio on the cable below an acceptable level.

A time domain reflectometer (TDR) has proven valuable in locating shorts, missing terminators, and even transceivers in the "on" state (since they have a low output impedance while transmitting). It is useful to have a reasonably accurate record of how the cable is routed, so that the linear distance from the TDR may be used to determine the approximate location of the failing component.

Segmenting the network is a useful technique when a TDR is not available, assuming that connectors have been installed in the cable at reasonable intervals. The network is broken into segments and each segment terminated at the point of separation. The segment that does not work contains the fault. (Multiple Ethernet systems interconnected by gateways have the property that a failure is localized to one network.)

5. Experience with an operational design

The prototype Ethernet system has served for several years as the dominant local network in a nationwide Pup internet used regularly within a number of Xerox research and development communities. The internet presently consists of about 25 Ethernet systems interconnected by 20 or so gateways and various long-haul transmission facilities. Over 1000 hosts (mostly Altos and other personal minicomputers) are connected to these networks.

Practical experience with the Ethernet system has shown the strengths and weaknesses of the present design. This section summarizes our experience and highlights those aspects of the design that could be improved.

5.1 Performance

The analytical model of CSMA/CD performance under *heavy load* and *overload* conditions, first presented in [Metcalfe & Boggs, 1976], has now been verified experimentally [Shoch & Hupp, 1979]. The network remains stable, its capacity is shared fairly among the contending hosts, and efficiency exceeds 97 per cent when traffic consists of relatively large packets (about 4000 bits).

Of at least equal importance is the performance of higher-level applications under *typical* conditions. Practical experience has shown that the consequences of certain design and engineering choices, such as those presented previously, totally dominate the undesirable effects predicted by analysis of an overloaded Ethernet system. (For example, in [Almes & Lazowska, 1979] it has been shown that the "binary exponential backoff" algorithm used for contention resolution in the Ethernet gives rise to a rather curious last-in-first-out acquisition of the channel. This phenomenon has indeed been observed under extreme overload conditions, but is of no consequence during normal operation.)

Typical conditions. It is useful to characterize what "typical" conditions are. Extensive measurements have been performed on one operational Ethernet system to which approximately 120 computers are connected. These machines include personal computers, servers, time-sharing systems, and gateways to other parts of the Pup internet. Users of these machines use the network regularly for file transfer, electronic mail, bootstrap loading, software distribution, and many other purposes.

In a typical 24-hour period, 2.2 million packets are carried in this network, totalling approximately 300 million bytes. For a 3-megabit per second channel, this amounts to only 0.9 percent utilization of the available bandwidth. But the utilization has high variance, and observed utilization has been as high as 3.6 percent for one hour, 17 percent for one minute, and 37 percent for one second. On the average, only 0.79 percent of all packets are delayed due to deference, and less than 0.03 percent are involved in collisions.

Thus, the existing system could support a substantial increase in traffic without a noticeable increase in latency. This is as it should be. While it is reassuring to know that the Ethernet system will remain stable under overload conditions, it would be bad practice to operate a local network regularly at more than a fraction of its peak capacity, especially in view of the high variations in utilization.

End-to-end performance. In the environment served by Pup, the most important metric for evaluating the Ethernet system is the performance delivered to the communicating end processes. In view of the negligible impact on overall performance due to contention for the shared channel, this performance is primarily a function of other components of the system. In the prototype implementation, there are two principal limitations on end-to-end performance.

Receiver turnaround. The inability of an Alto interface to receive consecutive packets or to receive a packet that immediately follows that Alto's own transmission has serious performance consequences. Many packets are missed simply because the receiver is not turned on at the time a packet begins to arrive. This occurs most frequently in servers and gateways, which may be communicating with several hosts at the same time; but it can also occur in any pair of hosts that happen to be sending to each other simultaneously.

Timeout-driven higher-level protocols can and do cope with lost packets, but when the rate of loss becomes too great the end-to-end throughput drops dramatically. This effect completely swamps the degradation caused by contention and damaged packets. For most applications this is still not noticeable, but for some it is serious enough that we have found it necessary to tune higher-level protocol implementations so as to reduce the probability of losing packets due to turnaround latency. This is a serious intrusion upon what is supposed to be a network-independent protocol.

This problem is due to the Alto controller hardware, which is half-duplex and requires restart by software or microcode after transmission or reception of a packet. The present design was arrived at after consideration of maximum chip count (approximately 75 TTL MSI ICs, using technology available in 1973). New designs should be capable of receiving consecutive packets. Fortunately, advances in IC technology have now made this possible at low cost.

Software overhead. End-to-end throughput is primarily limited by per-packet processing overhead in the software. For example, present implementations of the Pup Byte Stream Protocol, written in a high-level language (BCPL or Mesa) and running on the Alto, are capable of maintaining a sequenced, flow-controlled data stream at the rate of about 600 kilobits per second, at which point both sending and receiving computers are CPU-bound. While this performance is fairly respectable considering the modest processing capability of an Alto, it is far below the capacity of the Ethernet channel.

Reducing per-packet software overhead may be accomplished by designing "smart" controllers that are better tuned to the needs of higher-level protocols. Sources of overhead include packet completion interrupt handling, Pup encapsulation and decapsulation, collection of packets from and delivery to the correct processes, and higher-level protocol overhead such as sequencing, generation and processing of acknowledgments, and so on. Some of these tasks could be taken over by Alto microcode or (in other designs) by a dedicated front-end microcomputer.

The danger in this approach is that higher-level protocol functions may become inappropriately embedded in the controller, thereby blurring the levels of protocol and reducing the flexibility available to the end communicating processes. To avoid this, one should attempt to provide such functions in the form of operations *accessible from software* as opposed to ones *permanently interposed* between the network and the end processes. (This observation applies equally well to other operations traditionally thought of as being functions of a communications interface—for example, encryption [Needham & Schroeder, 1978].)

5.2 Reliability

One of the major design goals of the Ethernet system was that it be extremely reliable, in the sense that the shared communications channel should be immune from catastrophic failure. Failures that affect an individual host or that cause isolated packets to be lost or damaged are of secondary importance. This goal (among others) led to the passive bus structure in which most of the components—in particular, all the active ones—are associated with the host computers rather than being part of the channel itself, thereby reducing the likelihood that a single malfunction will cause total failure of the network.

This approach has been very successful in practice. Failures of an entire Ethernet system are quite rare; for example, the heavily-populated network described previously (120 hosts) suffers outages only a few times per year, and only for a few minutes at a time while the problem is located and corrected. Failures that affect individual hosts or that increase the packet error rate are, of course, more frequent. It is worth highlighting some of the most common malfunctions.

Total failures. The usual causes of catastrophic failure are incorrectly installed taps, shorted or malfunctioning transceivers, and broken controllers that transmit data indefinitely.

A bad tap or shorted transceiver is potentially the most troublesome problem to deal with, since locating and correcting it might require inspecting long stretches of the coaxial cable. (This concern has motivated network structures such as the "star-shaped ring", which features centralized fault isolation [Saltzer & Pogran, 1979].) In practice, however, such failures are almost invariably associated with physical activity such as installing a new tap or transceiver, and are noticed and corrected immediately.

The most serious failure that has been encountered to date occurred when lightning struck near a pair of buildings that had an Ethernet cable strung underground between them. The input

transistors in a dozen or so transceivers became fused. Finding all the shorted transceivers took several hours. Partitioning the network made it apparent that there were problems in all sections. A time domain reflectometer (TDR) was helpful in locating the damaged transceivers among the 100 or so units connected to that network.

The likely cause of failure was a shift in ground potentials between the two buildings, which was coupled into the cable shield through several tap blocks that were found to be resting against grounded cable trays. The voltage drop along the length of the shield was not matched by a corresponding drop along the center conductor, and consequently half of the voltage present between the building grounds was also present between the center conductor and shield of the coax cable. This voltage damaged the input transistors in the transceivers. An improved design would include insulated tap blocks and connector housings as well as current-limiting resistors in the transceiver input circuits.

Occasionally a controller malfunctions in such a way as to transmit continuous, correctly phase-encoded data. (As explained earlier, the transceiver prevents the controller from simply jamming the channel "on".) Failures can also be caused by malfunctioning software; for example, a program that repeatedly transmits broadcast packets can have a devastating effect on the performance of every host on the network.

Uncontrolled pollution of this nature can be prevented by introduction of a watchdog timer that electrically disconnects the transceiver from the bus unless periodically reset by the software. Such a scheme has been used in the Queen Mary College ENET and CNET in a manner that also verifies the proper functioning of the controller and low-level software [West & Davison, 1978].

Isolated failures. The frequency of packets lost due to interface-detected errors (incorrect CRC or termination at other than a word boundary) is about one in 6000 when averaged over many machines. This is so small as to have negligible effect on overall performance. Furthermore, an overwhelming majority of these errors are attributable to defects in the Alto Ethernet interface rather than to corruption on the wire. A redesigned interface, which features digital rather than analog phase decoding, has reduced the frequency of damaged packets to less than one in 2 million [Shoch & Hupp, 1979].

The electrical signalling used on the cable is such that the sync bit of a packet barely passes the DC receiver threshold after traversing one kilometer of cable, due to the stretching of pulse rise-times [Wigington & Nahman, 1957]. The average DC level is achieved only after several bit times. This results in occasional failure of a receiver to acquire correct bit phase. A worthwhile modification to the signalling protocol would be to precede each packet's sync bit with a fairly long preamble. This would permit higher speed operation while still using a DC receiver threshold. It would also permit phase decoding by means of a phase locked loop, which requires more than one bit time to lock onto the signal.

Higher-level protocols are resilient enough to mask rather gross hardware failures. For example, in a few instances a host's carrier sense circuit has failed, causing its transmitter not to

defer to an ongoing transmission but rather to collide with it. These and similar failures have negligible effects on end-to-end communication—even in the machines that have failed—and are detected only by careful network monitoring and measurement.

In the same vein, a host with a marginal transceiver or controller is sometimes unable to communicate with a particular set of other hosts but has no difficulty communicating with the rest. This is due primarily to variations in analog components.

Detecting this type of failure requires attempting communication between all n^2 pairs of hosts in the network. This may be done by a distributed diagnostic program, called a "worm", which runs in several Altos at a time and determines whether they can communicate with each other. Each instance of the program periodically finds another idle Alto, bootstrap-loads that Alto with a copy of itself, and destroys itself. As the worm crawls around the network, it reports its findings to a server which maintains a record of the network's logical connectivity.

6. Conclusion

The Ethernet system design has been successful in the role intended for it: a high-performance local transport mechanism for an internetwork architecture. The simplicity of the basic design, combined with careful engineering of the critical components, has resulted in great flexibility and high reliability. We have attempted to highlight the important positive and negative features of the prototype implementation and to assess their impact on an operational local network.

Acknowledgments

The Ethernet system was originally designed and constructed by Bob Metcalfe and Dave Boggs. The traffic and error statistics summarized here were the result of extensive measurements performed by John Shoch and Jon Hupp. Numerous other people in Xerox PARC and SDD have contributed to the success of the Ethernet.

References

[Abramson, 1977]
N. Abramson, The Throughput of Packet Broadcasting Channels, *IEEE Transactions on Communications*, vol. Com-25 no. 1, January 1977.

[Agrawala *et al.* 1977]
A. K. Agrawala, R. M. Bryant, and J. Agre, Analysis of an Ethernet-like Protocol, *Proceedings of the Computer Networking Symposium*, IEEE(CS)/NBS, Gaithersburg, December 1977.

[Almes & Lazowska, 1979]
G. T. Almes and E. D. Lazowska, *The Behavior of Ethernet-like Computer Communications Networks*, Technical Report no. 79-05-01, Dept. of Computer Science, University of Washington, April 1979.

[Boggs *et al*, 1980]
David R. Boggs, John F. Shoch, Edward A. Taft, and Robert M. Metcalfe, Pup: An Internetwork Architecture, *IEEE Transactions on Communications*, vol. Com-28 no. 4, April 1980, to appear; also available as Xerox PARC technical report CSL-79-10.

[Cerf & Kahn, 1974]
Vinton Cerf and Robert Kahn, A Protocol for Packet Network Interconnection, *IEEE Transactions of Communication*, vol. Com-22 no. 5, May 1974.

[Du Pont, 1979]
Plenum Cable of TEFLON FEP, E. I. Du Pont De Nemours & Co., Inc., Plastic Products & Resins Department, Wilmington, 1979.

[McQuillan, 1979]
John M. McQuillan, Local Network Technology and the Lessons of History, *Local Area Communications Network Symposium*, Boston, May 1979.

[Metcalfe, 1973]
Robert M. Metcalfe, *Packet Communication*, Harvard Ph.D. thesis, M.I.T. Project MAC TR-114, December 1973.

[Metcalfe & Boggs, 1976]
Robert M. Metcalfe and David R. Boggs, Ethernet: Distributed Packet Switching for Local Computer Networks, *Communications of the ACM*, vol. 19 no. 7, July 1976.

[Needham & Schroeder, 1978]
Roger Needham and Michael Schroeder, Using Encryption for Authentication in Large Networks of Computers, *Communications of the ACM*, vol. 21 no. 12, December 1978.

[Saltzer & Pogran, 1979]
Jerome H. Saltzer and Kenneth T. Pogran, A Star-shaped Ring Network with High Maintainability, *Local Area Communications Network Symposium*, Boston, May 1979.

[Shoch, 1979]
John F. Shoch, *Design and Performance of Local Computer Networks*, Stanford Ph.D. thesis, University Microfilms, August 1979.

[Shoch & Hupp, 1979]
John F. Shoch and Jon A. Hupp, Performance of an Ethernet Local Network—a Preliminary Report, *Local Area Communications Network Symposium*, Boston, May 1979.

[Sunshine, 1977]
Carl Sunshine, Interconnection of Computer Networks, *Computer Networks*, vol. 1, 1977.

[Thacker *et al.*, 1979]
C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs, Alto: A Personal Computer, *Computer Structures: Readings and Examples* (Siewiorek, Bell, and Newell, eds.), 1979, to appear; also available as Xerox PARC technical report CSL-79-11.

[Tobagi & Hunt, 1979]
F. A. Tobagi and V. B. Hunt, Performance Analysis of Carrier Sense Multiple Access with Collision Detection, *Local Area Communications Network Symposium,* Boston, May 1979.

[West & Davison, 1978]
Anthony West and Allan Davison, *CNET—a Cheap Network for Distributed Computing,* TR-120, Computer System Laboratory, Queen Mary College, March 1978.

[Wigington & Nahman, 1957]
R. L. Wigington and N. S. Nahman, Transient Analysis of Coaxial Cables Considering Skin Effect, *Proceedings of the IRE,* vol. 45, February 1957.