Computer Systems
G. Bell, S. Fuller and D. Siewiorek, Editors

Ethernet: Distributed Packet Switching for Local Computer Networks

Robert M. Metcalfe and David R. Boggs
Xerox Palo Alto Research Center

Ethernet is a branching broadcast communication system for carrying digital data packets among locally distributed computing stations. The packet transport mechanism provided by Ethernet has been used to build systems which can be viewed as either local computer networks or loosely coupled multiprocessors. An Ethernet's shared communication facility, its Ether, is a passive broadcast medium with no central control. Coordination of access to the Ether for packet broadcasts is distributed among the contending transmitting stations using controlled statistical arbitration. Switching of packets to their destinations on the Ether is distributed among the receiving stations using packet address recognition. Design principles and implementation are described, based on experience with an operating Ethernet of 100 nodes along a kilometer of coaxial cable. A model for estimating performance under heavy loads and a packet protocol for error controlled communication are included for completeness.

Key Words and Phrases: computer networks, packet switching, multiprocessing, distributed control, distributed computing, broadcast communication, statistical arbitration

CR Categories: 3.81, 4.32, 6.35

Author's present addresses: R.M. Metcalfe, Transaction Technology, Inc., 10880 Wilshire Boulevard, Los Angeles, CA 94304; D. Boggs, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304.

1. Background

One can characterize distributed computing as a spectrum of activities varying in their degree of decentralization, with one extreme being remote computer networking and the other extreme being multiprocessing. Remote computer networking is the loose interconnection of previously isolated, widely separated, and rather large computing systems. Multiprocessing is the construction of previously monolithic and serial computing systems from increasingly numerous and smaller pieces computing in parallel. Near the middle of this spectrum is local networking, the interconnection of computers to gain the resource sharing of computer networking and the parallelism of multiprocessing.

The separation between computers and the associated bit rate of their communication can be used to divide the distributed computing spectrum into broad activities. The product of separation and bit rate, now about 1 gigabit-meter per second (1 Gbmps), is an indication of the limit of current communication technology and can be expected to increase with time:

Activity	Separation	Bit rate
Remote networks	> 10 km	< .1 Mbps
Local networks	10–.1 km	.1–10 Mbps
Multiprocessors	< .1 km	> 10 Mbps

1.1 Remote Computer Networking

Computer networking evolved from telecommunications *terminal-computer* communication, where the object was to connect remote terminals to a central computing facility. As the need for *computer-computer* interconnection grew, computers themselves were used to provide communication [2, 4, 29]. Communication *using* computers as packet switches [15–21, 26] and communications *among* computers for resource sharing [10, 32] were both advanced by the development of the Arpa Computer Network.

The Aloha Network at the University of Hawaii was originally developed to apply packet radio techniques for communication between a central computer and its terminals scattered among the Hawaiian Islands [1, 2]. Many of the terminals are now minicomputers communicating among themselves using the Aloha Network's Menehune as a packet switch. The Menehune and an Arpanet Imp are now connected, providing terminals on the Aloha Network access to computing resources on the U.S. mainland.

Just as computer networks have grown across continents and oceans to interconnect major computing facilities around the world, they are now growing down corridors and between buildings to interconnect minicomputers in offices and laboratories [3, 12, 13, 14, 35].

1.2 Multiprocessing

Multiprocessing first took the form of connecting an I/O controller to a large central computer; IBM's Asp is a

classic example [29]. Next, multiple central processors were connected to a common memory to provide more power for compute-bound applications [33]. For certain of these applications, more exotic multiprocessor architectures such as Illiac IV were introduced [5].

More recently minicomputers have been connected in multiprocessor configurations for economy, reliability, and increased system modularity [24, 36]. The trend has been toward decentralization for reliability; loosely coupled multiprocessor systems depend less on shared central memory and more on *thin wires* for interprocess communication with increased component isolation [18, 26]. With the continued thinning of interprocessor communication for reliability and the development of distributable applications, multiprocessing is gradually approaching a local form of distributed computing.

1.3 Local Computer Networking

Ethernet shares many objectives with other local networks such as Mitre's Mitrix, Bell Telephone Laboratory's Spider, and U.C. Irvine's Distributed Computing System (DCS) [12, 13, 14, 35]. Prototypes of all four local networking schemes operate at bit rates between one and three megabits per second. Mitrix and Spider have a central minicomputer for switching and bandwidth allocation, while DCS and Ethernet use distributed control. Spider and DCS use a ring communication path, Mitrix uses off-the-shelf CATV technology to implement two one-way busses, and our experimental Ethernet uses a branching two-way passive bus. Differences among these systems are due to differences among their intended applications, differences among the cost constraints under which trade-offs were made, and differences of opinion among researchers.

Before going into a detailed description of Ethernet, we offer the following overview (see Figure 1).

2. System Summary

Ethernet is a system for local communication among computing stations. Our experimental Ethernet uses tapped coaxial cables to carry variable length digital data packets among, for example, personal minicomputers, printing facilities, large file storage devices, magnetic tape backup stations, larger central computers, and longer-haul communication equipment.

The shared communication facility, a branching Ether, is passive. A station's Ethernet interface connects bit-serially through an interface cable to a transceiver which in turn taps into the passing Ether. A packet is broadcast onto the Ether, is heard by all stations, and is copied from the Ether by destinations which select it according to the packet's leading address bits. This is broadcast packet switching and should be distinguished from store-and-forward packet switching, in which routing is performed by intermediate processing elements. To handle the demands of growth, an Ethernet can be extended using packet repeaters for signal regeneration, packet filters for traffic localization, and packet gateways for internetwork address extension.

Control is completely distributed among stations, with packet transmissions coordinated through statistical arbitration. Transmissions initiated by a station defer to any which may already be in progress. Once started, if interference with other packets is detected, a transmission is aborted and rescheduled by its source station. After a certain period of interference-free transmission, a packet is heard by all stations and will run to completion without interference. Ethernet controllers in colliding stations each generate random retransmission intervals to avoid repeated collisions. The mean of a packet's retransmission intervals is adjusted as a function of collision history to keep Ether utilization near the optimum with changing network load.

Even when transmitted without source-detected interference, a packet may still not reach its destination without error; thus, packets are delivered *only with high probability*. Stations requiring a residual error rate lower than that provided by the bare Ethernet packet transport mechanism must follow mutually agreed upon packet protocols.

3. Design Principles

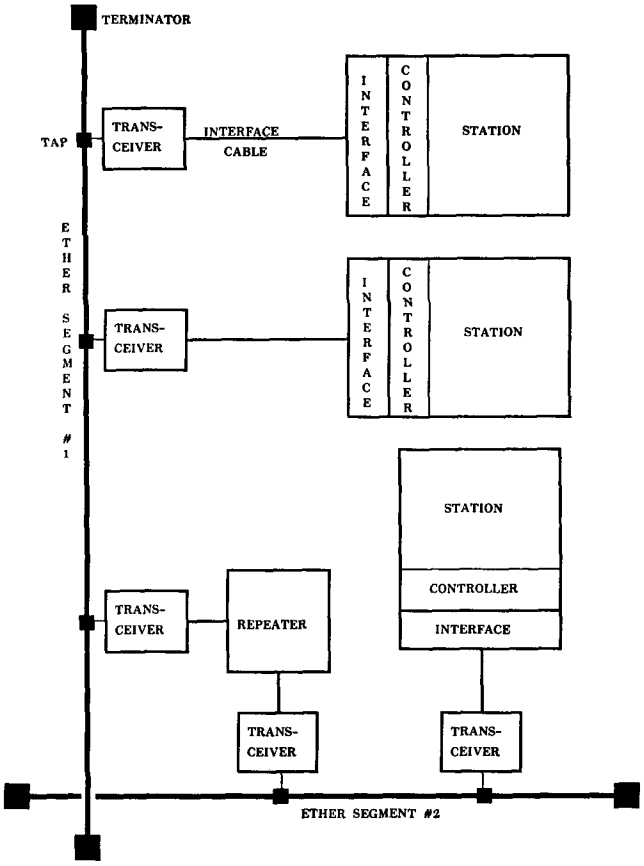
Our object is to design a communication system which can grow smoothly to accommodate several buildings full of personal computers and the facilities needed for their support.

Like the computing stations to be connected, the communication system must be inexpensive. We choose to distribute control of the communications facility among the communicating computers to eliminate the reliability problems of an active central controller, to avoid creating a bottleneck in a system rich in parallelism, and to reduce the fixed costs which make small systems uneconomical.

Ethernet design started with the basic idea of packet collision and retransmission developed in the Aloha Network [1]. We expected that, like the Aloha Network, Ethernets would carry bursty traffic so that conventional synchronous time-division multiplexing (STDM) would be inefficient [1, 2, 21, 26]. We saw promise in the Aloha approach to distributed control of radio channel multiplexing and hoped that it could be applied effectively with media suited to local computer communication. With several innovations of our own, the promise is realized.

Ethernet is named for the historical *luminiferous ether* through which electromagnetic radiations were once alleged to propagate. Like an Aloha radio transmitter, an Ethernet transmitter broadcasts completely-addressed transmitter-synchronous bit sequences called packets onto the Ether and hopes that they are heard by

Fig. 1. A two-segment Ethernet.



the intended receivers. The Ether is a logically passive medium for the propagation of digital signals and can be constructed using any number of media including coaxial cables, twisted pairs, and optical fibers.

3.1 Topology

We cannot afford the redundant connections and dynamic routing of store-and-forward packet switching to assure reliable communication, so we choose to achieve reliability through simplicity. We choose to make the shared communication facility passive so that the failure of an active element will tend to affect the communications of only a single station. The layout and changing needs of office and laboratory buildings leads us to pick a network topology with the potential for convenient incremental extention and reconfiguration with minimal service disruption.

The topology of the Ethernet is that of an unrooted tree. It is a *tree* so that the Ether can branch at the entrance to a building's corridor, yet avoid multipath interference. There must be only one path through the Ether between any source and destination; if more than one path were to exist, a transmission would interfere with itself, repeatedly arriving at its intended destination having travelled by paths of different length. The Ether is *unrooted* because it can be extended from any of its points in any direction. Any station wishing to join an Ethernet taps into the Ether at the nearest convenient point.

Looking at the relationship of interconnection and control, we see that Ethernet is the dual of a star network. Rather than *distributed* interconnection through many separate links and *central* control in a switching node, as in a star network, the Ethernet has *central* interconnection through the Ether and *distributed* control among its stations.

Unlike an Aloha Network, which is a star network with an outgoing broadcast channel and an incoming multi-access channel, an Ethernet supports many-to-many communication with a single broadcast multi-access channel.

3.2 Control

Sharing of the Ether is controlled in such a way that it is not only possible but probable that two or more stations will attempt to transmit a packet at roughly the same time. Packets which overlap in time on the Ether are said to *collide*; they interfere so as to be unrecognizable by a receiver. A station recovers from a detected collision by abandoning the attempt and retransmitting the packet after some dynamically chosen random time period. Arbitration of conflicting transmission demands is both distributed and statistical.

When the Ether is largely unused, a station transmits its packets at will, the packets are received without error, and all is well. As more stations begin to transmit, the rate of packet interference increases. Ethernet controllers in each station are built to adjust the mean retransmission interval in proportion to the frequency of collisons; sharing of the Ether among competing station-station transmissions is thereby kept near the optimum [20, 21].

A degree of cooperation among the stations is required to share the Ether equitably. In demanding applications certain stations might usefully take transmission priority through some systematic violation of equity rules. A station could usurp the Ether by not adjusting its retransmission interval with increasing traffic or by sending very large packets. Both practices are now prohibited by low-level software in each station.

3.3 Addressing

Each packet has a source and destination, both of which are identified in the packet's header. A packet placed on the Ether eventually propagates to all stations. Any station can copy a packet from the Ether into its local memory, but normally only an active destination station matching its address in the packet's header will do so as the packet passes. By convention, a zero destination address is a wildcard and matches all addresses; a packet with a destination of zero is called a *broadcast packet*.

3.4 Reliability

An Ethernet is probabilistic. Packets may be lost due to interference with other packets, impulse noise on the

Ether, an inactive receiver at a packet's intended destination, or purposeful discard. Protocols used to communicate through an Ethernet must assume that packets will be received correctly at intended destinations *only with high probability*.

An Ethernet gives its *best efforts* to transmit packets successfully, but it is the responsibility of processes in the source and destination stations to take the precautions necessary to assure reliable communication of the quality they themselves desire [18, 21]. Recognizing the costliness and dangers of promising "error-free" communication, we refrain from guaranteeing reliable delivery of any single packet to get both economy of transmission and high reliability averaged over many packets [21]. Removing the responsibility for reliable communication from the packet transport mechanism allows us to tailor reliability to the application and to place error recovery where it will do the most good. This policy becomes more important as Ethernets are interconnected in a hierarchy of networks through which packets must travel farther and suffer greater risks.

3.5 Mechanisms

A station connects to the Ether with a *tap* and a *transceiver*. A tap is a device for physically connecting to the Ether while disturbing its transmission characteristics as little as possible. The design of the transceiver must be an exercise in paranoia. Precautions must be taken to insure that likely failures in the transceiver or station do not result in pollution of the Ether. In particular, removing power from the transceiver should cause it to disconnect from the Ether.

Five mechanisms are provided in our experimental Ethernet for reducing the probability and cost of losing a packet. These are (1) carrier detection, (2) interference detection, (3) packet error detection, (4) truncated packet filtering, and (5) collision consensus enforcement.

3.5.1 Carrier detection. As a packet's bits are placed on the Ether by a station, they are phase encoded (like bits on a magnetic tape), which guarantees that there is at least one transition on the Ether during each bit time. The passing of a packet on the Ether can therefore be detected by listening for its transitions. To use a radio analogy, we speak of the presence of *carrier* as a packet passes a transceiver. Because a station can sense the carrier of a passing packet, it can delay sending one of its own until the detected packet passes safely. The Aloha Network does not have carrier detection and consequently suffers a substantially higher collision rate. Without carrier detection, efficient use of the Ether would decrease with increasing packet length. In Section 6 below, we show that with carrier detection, Ether efficiency increases with increasing packet length.

With carrier detection we are able to implement *deference*: no station will start transmitting while hearing carrier. With deference comes *acquisition*: once a packet transmission has been in progress for an Ether end-to-end propagation time, all stations are hearing carrier and are deferring; the Ether has been acquired and the transmission will complete without an interfering collision.

With carrier detection, collisions should occur only when two or more stations find the Ether silent and begin transmitting simultaneously: within an Ether end-to-end propagation time. This will almost always happen immediately after a packet transmission during which two or more stations were deferring. Because stations do not now randomize after deferring, when the transmission terminates, the waiting stations pile on together, collide, randomize, and retransmit.

3.5.2 Interference detection. Each transceiver has an interference detector. Interference is indicated when the transceiver notices a difference between the value of the bit it is receiving from the Ether and the value of the bit it is attempting to transmit.

Interference detection has three advantages. First, a station detecting a collision knows that its packet has been damaged. The packet can be scheduled for retransmission immediately, avoiding a long acknowledgment timeout. Second, interference periods on the Ether are limited to a maximum of one round trip time. Colliding packets in the Aloha Network run to completion, but the truncated packets resulting from Ethernet collisions waste only a small fraction of a packet time on the Ether. Third, the frequency of detected interference is used to estimate Ether traffic for adjusting retransmission intervals and optimizing channel efficiency.

3.5.3 Packet error detection. As a packet is placed on the Ether, a checksum is computed and appended. As the packet is read from the Ether, the checksum is recomputed. Packets which do not carry a consistent checksum are discarded. In this way transmission errors, impulse noise errors, and errors due to undetected interference are caught at a packet's destination.

3.5.4 Truncated packet filtering. Interference detection and deference cause most collisions to result in *truncated packets* of only a few bits; colliding stations detect interference and abort transmission within an Ether round trip time. To reduce the processing load that the rejection of such obviously damaged packets would place on listening station software, truncated packets are filtered out in hardware.

3.5.5 Collision consensus enforcement. When a station determines that its transmission is experiencing interference, it momentarily jams the Ether to insure that all other participants in the collision will detect interference and, because of deference, will be forced to abort. Without this *collision consensus enforcement* mechanism, it is possible that the transmitting station which would otherwise be the last to detect a collision might not do so as the other interfering transmissions successively abort and stop interfering. Although the packet may look good to that last transmitter, different path lengths

between the colliding transmitters and the intended receiver will cause the packet to arrive damaged.

4. Implementation

Our choices of 1 kilometer, 3 megabits per second, and 256 stations for the parameters of an experimental Ethernet were based on characteristics of the locally distributed computer communication environment and our assessments of what would be marginally achievable; they were certainly not hard restrictions essential to the Ethernet concept.

We expect that a reasonable maximum network size would be on the order of 1 kilometer of cable. We used this working number to choose among Ethers of varying signal attenuation and to design transceivers with appropriate power and sensitivity.

The dominant station on our experimental Ethernet is a minicomputer for which 3 megabits per second is a convenient data transfer rate. By keeping the peak rate well below that of the computer's path to main memory, we reduce the need for expensive special-purpose packet buffering in our Ethernet interfaces. By keeping the peak rate as high as is convenient, we provide for larger numbers of stations and more ambitious multiprocessing communications applications.

To expedite low-level packet handling among 256 stations, we allocate the first 8-bit byte of the packet to be the destination address field and the second byte to be the source address field (see Figure 2). 256 is a number small enough to allow each station to get an adequate share of the available bandwidth and approaches the limit of what we can achieve with current techniques for tapping cables. 256 is only a convenient number for the lowest level of protocol; higher levels can accomodate extended address spaces with additional fields inside the packet and software to interpret them.

Our experimental Ethernet implementation has four major parts: the Ether, transceivers, interfaces, and controllers (see Figure 1).

4.1 Ether

We chose to implement our experimental Ether using low-loss coaxial cable with off-the-shelf CATV taps and connectors. It is possible to mix Ethers on a single Ethernet; we use a smaller-diameter coax for convenient connection within station clusters and a larger-diameter coax for low-loss runs between clusters. The cost of coaxial cable Ether is insignificant relative to the cost of the distributed computing systems supported by Ethernet.

4.2 Transceivers

Our experimental transceivers can drive a kilometer of coaxial cable Ether tapped by 256 stations transmitting at 3 megabits per second. The transceivers can *endure* (i.e. work after) sustained direct shorting, improper termination of the Ether, and simultaneous drive by all 256 stations; they can *tolerate* (i.e. work during) ground differentials and everyday electrical noise, from typewriters or electric drills, encountered when stations are separated by as much as a kilometer.

An Ethernet transceiver attaches directly to the Ether which passes by in the ceiling or under the floor. It is powered and controlled through five twisted pairs in an interface cable carrying transmit data, receive data, interference detect, and power supply voltages. When unpowered, the transceiver disconnects itself electrically from the Ether. Here is where our fight for reliability is won or lost; a broken transceiver can, but should not, bring down an entire Ethernet. A watchdog timer circuit in each transceiver attempts to prevent pollution of the Ether by shutting down the output stage if it acts suspiciously. For transceiver simplicity we use the Ether's base frequency band, but an Ethernet could be built to use any suitably sized band of a frequency division multiplexed Ether.

Even though our experimental transceivers are very simple and can tolerate only limited signal attenuation, they have proven quite adequate and reliable. A more sophisticated transceiver design might permit passive branching of the Ether and wider station separation.

4.3 Interface

An Ethernet interface serializes and deserializes the parallel data used by its station. There are a number of different stations on our Ethernet; an interface must be built for each kind.

Each interface is equipped with the hardware necessary to compute a 16-bit cyclic redundancy checksum (CRC) on serial data as it is transmitted and received. This checksum protects only against errors in the Ether and specifically not against errors in the parallel portions of the interface hardware or station. Higher-level software checksums are recommended for applications in which a higher degree of reliability is required.

A transmitting interface uses a packet buffer address and word count to serialize and phase encode a variable number of 16-bit words which are taken from the station's memory and passed to the transceiver, preceded by a start bit (called SYNC in Figure 2) and followed by the CRC. A receiving interface uses the appearance of carrier to detect the start of a packet and uses the SYNC bit to acquire bit phase. As long as carrier stays on, the interface decodes and deserializes the incoming bit stream depositing 16-bit words in a packet buffer in the station's main memory. When carrier goes away, the interface checks that an integral number of 16-bit words has been received and that the CRC is correct. The last word received is assumed to be the CRC and is not copied into the packet buffer.

These interfaces ordinarily include hardware for accepting only those packets with appropriate addresses in their headers. Hardware address filtering helps a station avoid burdensome software packet processing when

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.