Networks and Communications Consulting

21885 Bear Creek Way
Los Gatos CA 95030
(408) 395-5700
FAX (408) 395-1955
Internet: seifert@netcom.com

Technical Report

The Effect of Ethernet Behavior on Networks using High-Performance Workstations and Servers

Author: Rich Seifert
March 3, 1995

1. Introduction

Over the past 15 years, Ethernet has been become the most widely-used technology for interconnecting workstations, servers, and other computer equipment in local area networks. For most of its life, the capacity of Ethernet (10 Mb/s) far exceeded the data-moving capability of the attached equipment or the applications using the network. Throughput was typically limited by controller implementations, driver and protocol software, or host and application performance.

More recently, the emergence of high-performance Ethernet implementations, as well as high-performance hosts (both workstations and servers) have changed this situation. In certain environments, the performance of the Ethernet itself may become the limiting factor in user throughput. In addition, these high-performance environments can uncover some side effects of the fundamental Ethernet design that were previously unobserved with lighter loads and slower implementations.

This paper:

- ❑ Identifies certain observable idiosyncratic behaviors of high-performance networked systems,
- ❑ Shows how these behaviors can be accounted for by the fundamental Ethernet design,
- ❑ Corrects some widely-held misconceptions about Ethernet behavior, and
- ❑ Offers some potential solutions to the problems.

1.1. Problem Statement

Consider a user with an existing set of equipment on an Ethernet network. Over years of use, they have developed certain expectations regarding application performance, network statistics reports, and system error messages. They know what is “normal”, or typical for their working environment. If there is a significant change in any of these factors, it is usually considered a problem which attracts attention and demands resolution.

As in most environments, there is a continuing need to improve total system performance, due to new application requirements or increases in the number of users being served. Over time, this user will upgrade key components of the network, including servers, workstations, etc. as demanded by the organization’s needs and budget. This newer equipment (e.g., a new server) is capable of increased network throughput, due both to higher processing capabilities, and the use of new, high-performance Ethernet controller implementations.

Many early controllers were unable to sustain 100% load on a 10 Mb/s Ethernet; newer controllers with faster logic can easily transmit or receive data continuously at the full channel data rate.

The user would expect that the introduction of this new equipment would:

- Improve user application throughput,
- Improve network performance statistics, and
- Reduce error messages.

Interestingly, there are scenarios where exactly the opposite can happen. Performance can worsen, statistics can change (e.g., greatly increased collision counts), and error messages can appear that did not appear with the older equipment.

1.2. Reader Assumptions

This report is not a primer on networking, LANs or protocol behavior. The reader is expected to be familiar with:

- Computer networks and layered protocol architectures,
- Basic operation of the Ethernet, including the CSMA/CD algorithm,
- The physical configuration of Ethernets, including 10Base5, 10Base2, and 10Base-T systems

1.3. Network Assumptions and Applicability

This report describes and explains a number of behaviors, problems and idiosyncrasies observed in systems which use high-performance Ethernet implementations. These phenomena (and their solutions) are applicable to a wide variety of systems, running any of the protocol suites in common use today, including TCP/IP, DECnet™, AppleTalk™, ISO, etc. For the purpose of discussion and explanation, however, we will use the particular example of a client-server environment, using NFS (Network File System) over UDP/IP on a local Ethernet, as shown in Figure 1.

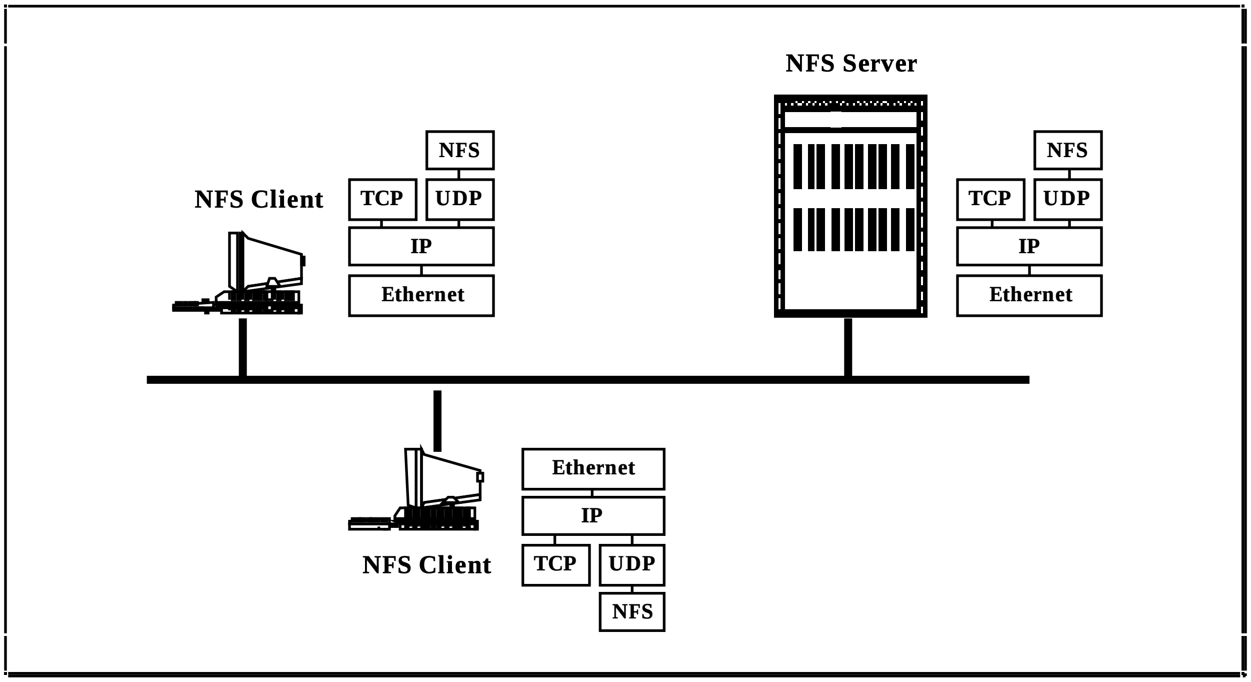

Figure 1: Target Environment

The scenarios discussed, and the solutions proposed, are applicable in any system where:

- ❑ A reliable high-layer protocol (typically Transport) provides a mechanism to recover from errors in the lower layers (i.e., error control), and
- ❑ The error control mechanism uses a Positive Acknowledgment and Retransmission strategy (PAR) with a retransmission timer significantly longer than the allowed window transmission time. Examples of such protocols include: NFS, TCP, ATP (AppleTalk™ Transaction Protocol), NSP (DECnet™ Network Services Protocol), and ISO 8073 (TP4).
- ❑ The underlying Network and Data Link layers provide a connectionless (datagram, or best-effort) delivery service. Examples include: IP, DDP (AppleTalk™ Datagram Delivery Protocol), DECnet™ Routing Protocol and ISO 8473 (CLNP) at the Network Layer. At the Data Link Layer, Ethernet and all IEEE 802 (and FDDI) LANs using IEEE 802.2 Class 1 service are connectionless.

These are extremely broad qualifications; i.e., this report applies to most LAN-based computer networks.

1.4 Terminology

This paper deals with phenomena occurring at many layers of the protocol architecture. Strictly speaking:

- ❑ A *frame* is the unit of data at the Data Link layer (e.g., an Ethernet frame),
- ❑ A *datagram* is the unit of data at the Network Layer (e.g., an IP datagram). Furthermore, some Network Layer protocols (e.g., IP) are capable of breaking a datagram into smaller units, known as *fragments*.

While it is more precise to use the terms *frame*, *datagram* and *fragment* where the specific meaning is applicable, in some cases the distinction is unimportant (e.g., higher layer protocols such as NFS do not care whether it did not get an acknowledgment due to a missing frame, datagram or fragment; it behaves the same in all cases. This paper uses the generic term *packet* when referring to situations equally applicable to frames, datagrams and fragments.

In addition, there is a distinction between a “lost” packet and a “discarded” packets. In a perfect world, every packet that is discarded can and should be accounted for. That is, even in the face of discarded packets, one should be able to “balance the books”. A packet which was truly lost, however, cannot be so accounted. A sender reports that it sent the packet correctly, but it never showed up at the receiver, and could therefore not be counted. From a higher-layer protocol perspective, the distinction is moot; e.g. NFS responds the same for both a discarded and a lost packet. In cases where the distinction is irrelevant, this paper uses the term “missing” to refer to a packet that was either discarded or lost.

2. Lower Application Throughput

As stated earlier, it is possible for application performance to actually degrade when changing from older systems to newer systems with greater capability. Key to an understanding of this problem scenario is an insight into the way in which the higher-layer protocol (in our example, NFS) operates.

2.1. Principles of NFS

NFS incorporates a relatively simple, reliable data transport mechanism. It provides both flow and error control in a manner avoiding the complexities of many other protocols, such as TCP.

NFS transfers blocks of data between peer entities across a network. The standard NFS block is 8 Kbytes (or less, if there is less than 8 Kbytes of data to be transferred). Each block has a unique identifier tag (ID) associated with it. The receiver of a block is required to acknowledge receipt of each block (using a short acknowledgment message). If the sender does not receive the acknowledgment within a specified time, it retransmits the original block. The retransmission timer is specified in NFS to be *at least* 700 ms; i.e., the failure of a block to be successfully transmitted or acknowledged will cause a retransmission 700 ms (or more) later.

A practical NFS implementation does not have infinite buffers, nor an infinite transmission window. If the underlying network has a high data rate (e.g., 10 Mb/s in Ethernet), NFS will generally run out of data to send (to a given receiver) before the expiration of the 700 ms retransmission timer. Thus, the failure of a block to be successfully transmitted and acknowledged will cause some period of "quiet time", when the sending station has run out of data to send and is waiting for the retransmission timer to expire before continuing. This is depicted in Figure 2.

While host implementations may vary, rarely will a host provide more than a few blocks per NFS connection on either read or write. Since the actual transmission time of an 8 Kbyte block on an Ethernet is < 7 ms (including protocol overhead, but not including protocol processing or backoff delay), the retransmission timer will always be much longer than the transmission window, and a significant "quiet time" will always be imposed in the event of a missing block or acknowledgment.

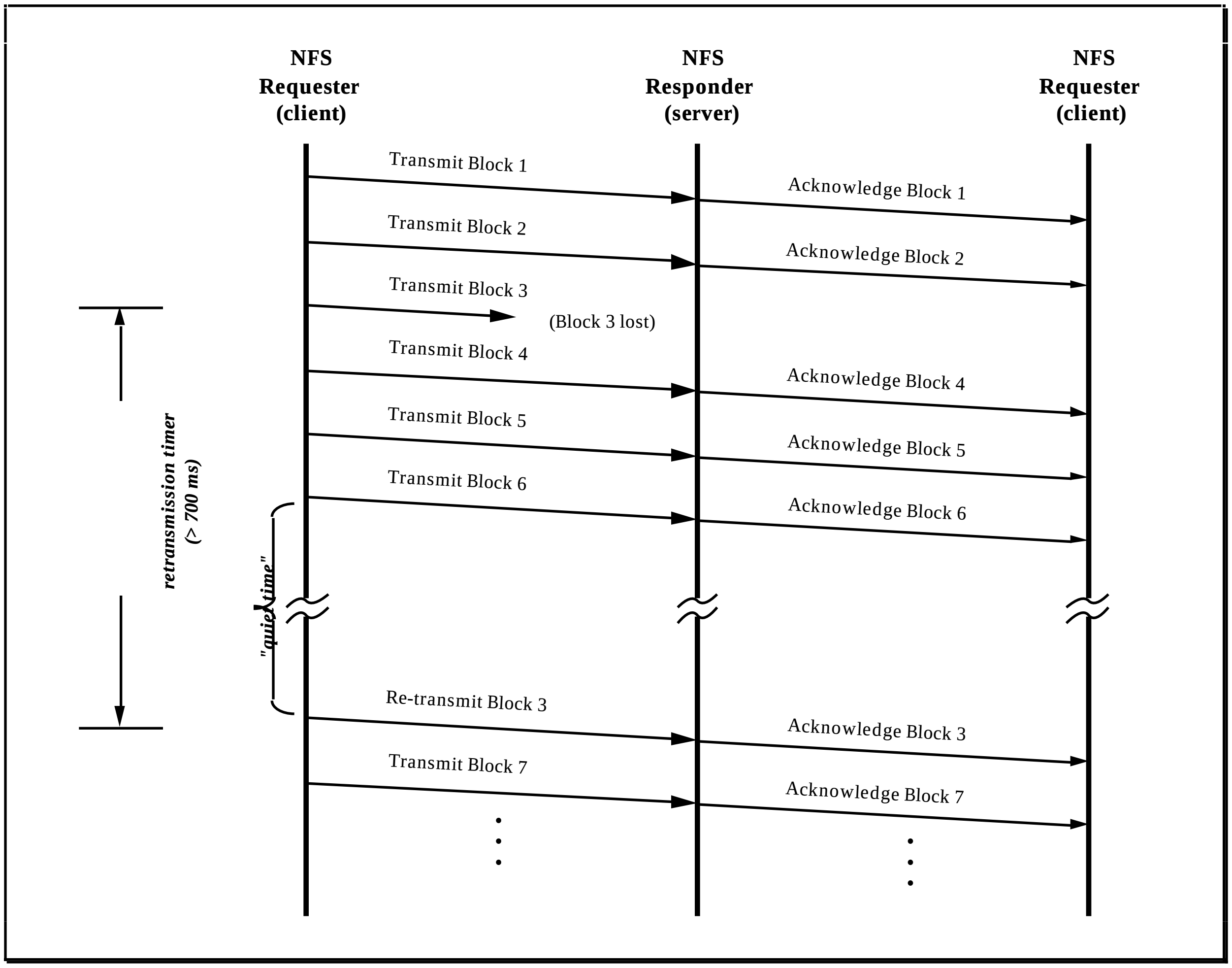

Figure 2: NFS Protocol Exchanges

This quiet time is effectively a *delay* imposed on the progress of NFS due to the failure to successfully transmit and acknowledge a block. This delay directly affects NFS performance and application throughput, and can be observed by most standard performance measurements. The more errors that occur, the more often this quiet time will be imposed, and the worse will be the measured NFS throughput.

2.2. MTU and Fragmentation

If the underlying network or LAN cannot carry 8 Kbyte messages directly (i.e., the Maximum Transmission Unit or MTU of the LAN is less than 8 Kbytes), then it is the responsibility of the lower layers to break the message into manageable pieces for transmission on the link. Typically, this is handled by the fragmentation and reassembly mechanism of IP. When running on an Ethernet, IP will fragment an 8 Kbyte NFS block into 6 IP datagrams, since the maximum payload of an Ethernet frame is 1500 bytes.

IP has no mechanism for recovering from missing packets, including missing fragments generated due to LAN MTU limitations. Thus, the loss or discard of even a single fragment will cause IP to be unable to reassemble the datagram, triggering an NFS timeout and retransmission.

2.3. Connectionless Underlying Service (UDP/IP/Ethernet)

The “problem” is that the underlying service (i.e., IP and Ethernet) is *connectionless;* no guarantees are made regarding successful delivery of data. The network makes a “best-effort” to deliver, but if a packet is missing it is the responsibility of the higher-layer protocol (NFS) to recover.

The reason for designing systems this way it that it greatly simplifies the design of Network and Data Link protocols and devices if there is no error control mechanism required. Controllers, bridges, routers and other devices can be made much simpler and therefore less expensive in a connectionless system. In addition, connectionless protocols tend to be more robust in the face of faults and errors. Most protocol suites and networked systems today use connectionless protocols at the underlying (Network and Data Link) layers.

Connectionless systems optimize the overall design for the case when there are no errors, sacrificing performance when errors do occur. As long as the probability of errors is low, systems using connectionless underlying services can operate efficiently and inexpensively.

2.4. The Effect of Missing Packets on NFS

In our connectionless environment, packets may be missing at a receiver for various reasons (e.g., checksum errors, excessive collisions, etc., as discussed later). A packet that is discarded or lost anywhere in the chain between sender and receiver (and back, for the acknowledgment) will cause an NFS time-out and retransmission, with its associated delay increase and performance loss.

2.5. Causes of Missing Packets

This section considers various causes of missing packets in a typical environment, with NFS running over an IP internetwork of LANs.

2.5.1. Channel Errors

No communications channel can be made error-free. Ultimately, noise in the physical medium will cause some residual level of errors, even in an otherwise ideal implementation.

Data Link protocols (including Ethernet) are designed to expect and detect such channel errors. Ethernet (and other LAN technologies) use a Cyclic Redundancy Checksum (CRC) to detect, with high probability, the occurrence of channel errors. Upon detection of an error, the receiving Data Link will discard the frame, and possibly increment a management counter to record the event.

A channel error thus results in a frame discard. Note that no connectionless link protocol (including Ethernet, FDDI, etc.) performs error correction or recovery from discarded frames. The design of the Data Link assumes that higher level protocols (such as NFS) will recover from frame discard if it needs to guarantee data delivery.

The probability of a channel error on a LAN is quite low. For the environment specified in [1], the worst-case channel error rate is $1x10^{-8}$. A typical error rate in a benign (office automation) environment is closer to $1x10^{-12}$. This means that for an Ethernet supporting NFS data transfers using maximum length Ethernet frames (1500 bytes) at a continuous 100% offered load, there should be on average one frame discarded (due to channel errors) per ~82 million frames. This translates to one discarded frame per 123 gigabytes transferred, or ~1 discarded frame per day at that sustained 100% load. Clearly, channel errors should not be a significant source of NFS retransmissions in a properly operating network. In fact, this is why the complexities involved in error detection and recovery were not built into LAN protocols in the first place.

2.5.2. Congestion and Errors in Bridges and Routers

An internetworking device (Data Link *Bridge* or Network Layer *Router*) can also discard packets. There are two mechanisms primarily responsible for packet discard in bridges and/or routers:

(1) Congestion

(2) Internal Errors

Congestion occurs when the internetworking device, which may be designed to handle network traffic under a wide variety of load conditions, has insufficient resources (e.g., buffers) to handle short-term peak load conditions.

This may be due to inherent limitations in the device. For example, the maximum worst-case back-to-back frame transmission rate on an Ethernet is 14,881 frames/second. While theoretically possible, no practical application will sustain this transmission rate in the steady-state. So it may be reasonable (as a conscious, price/performance tradeoff) to design an Ethernet bridge that cannot handle sustained load at that rate. If the bridge can receive 10,000 frames/second, maximum, there will be no perceived problems in the vast majority of application environments. However, if the traffic pattern on the Ethernet conspires to create a frame transmission rate beyond this limit, even for a short time, then the bridge will be unable to keep up with the incoming data and will ultimately discard frames.

Congestion can also occur even in internetworking devices designed to handle worst-case link loads. Consider the situation depicted in Figure 3, where all indicated links are 10 Mb/s Ethernets:

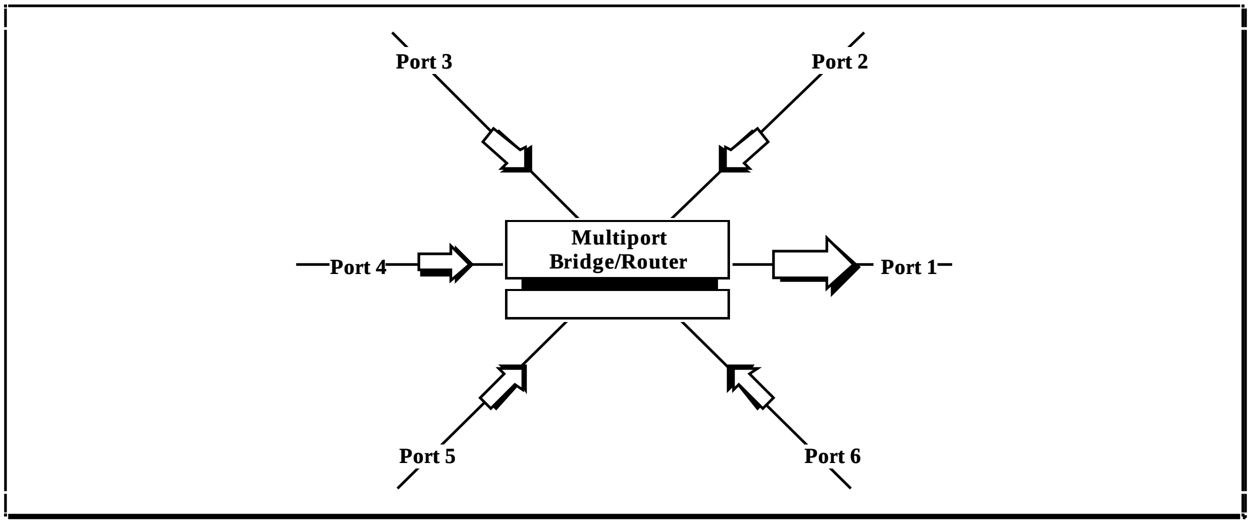

Figure 3: Network Congestion Scenario

Even if the router shown can sustain proper reception on all ports at full link speeds, if the traffic patterns are such that there is more traffic destined for a particular link than that link can carry, then ultimately the router's buffers will fill, and the device must begin discarding datagrams. This is a direct outcome of the lack of suitable flow and congestion control mechanisms in IP (and all connectionless network protocols).

2.5.3. Congestion in End Station Receivers

Similar to an internetworking device, an end station (client or server) LAN controller or host IP implementation may not be able to handle sustained offered load at the maximum link data rate. The station may be limited by buffer availability, host protocol processing limitations, or limitations in the LAN controller IC itself. If there is a short-term overload condition, the station will discard packets, requiring recovery by the higher-layer protocol (e.g., NFS).

In fact, the inability of an end station to sustain receiver performance at the load offered by the network at any layer up to the layer providing flow control, for any reason, can result in missing packets at the higher layer, necessitating time-outs and retransmissions.

2.5.4. Congestion in Senders

Although less obvious than receive overflow in end stations, it is also possible for a sending station (client or server) to congest its transmitter. Consider the scenario depicted in Figure 4:

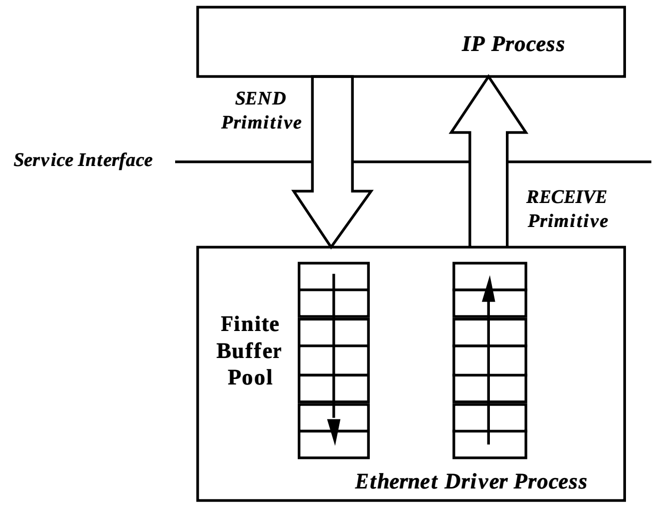

Figure 4: Partitioned Processing

There are separate-but-communicating processes for the IP protocol and the Ethernet device driver instantiations. If there is no mechanism for flow control across the service interface (i.e., between the communicating processes), then it is possible for a fast IP sender to attempt to send faster than the Ethernet driver can dispose of the requests. Ultimately, the driver will run out of buffers. If there is no way to communicate this information back to IP (and have IP throttle its transmissions as a result), IP will continue its attempts, and the driver will discard them. The driver, being unable to respond is discarding through ignorance or inability, not through an explicit mechanism.

From the perspective of NFS, however, this is just another instance of a missing packet, and results in the same time-out, retransmission and performance loss incurred by other sources of missing packets.

This syndrome can sometimes be observed by monitoring a device’s transmissions, and noting instances where only 5 of the 6 IP fragments required for an NFS block are actually transmitted by the station. Since the problem is due to buffer congestion, it will typically be the last of a burst of transmissions that will be lost. One important consideration is that in this scenario, the fragments are not transmitted and then discarded in the network, they are never transmitted at all.

2.5.4.1. LAN Congestion and Overload

Since the channel bandwidth in any LAN is finite, all LAN medium access control (MAC) protocols must provide some means of managing offered load to the channel. The method used in Ethernet is called Carrier Sense Multiple Access with Collision Detection (CSMA/CD) and is fully described in [1] and [2]. A key principle of this access method is that short-term overload conditions are managed by delaying frame delivery. That is, if there is a heavy demand, the collision and backoff algorithm implemented in all stations will increase the average delay for all transmitted frames.

Clearly, if the overload condition persists, frames cannot be delayed indefinitely. Ethernet stations make 16 attempts to transmit each frame. If the channel is so highly congested that a frame cannot be successfully transmitted in 16 attempts, the sending station discards the frame. Long-term (steady-state) overload such as this is indicative of an improperly configured or designed LAN; there is simply too much traffic for the available bandwidth.

However, short-term overloads can and do occur when there is a confluence of application network demand. Such an event may result in some stations discarding frames, requiring recovery by higher-layer protocols.

2.5.4.2. Capture Effect

Finally, there is a known idiosyncrasy of the Ethernet MAC algorithm, known as the *capture effect*, which can cause stations to discard frames because it *appears* that the LAN is congested as described above. The capture effect is an important source of discarded frames, and is described in §3, below, and further in [4].

2.6. Reducing NFS Retransmissions

We have seen that the error control mechanism in NFS (and other similar protocols) will impose a performance degradation in the face of missing packets. A packet discarded or lost anywhere in the chain will incur a time-out and retransmission, reducing application throughput.

Typically, there is no systematic problem with most networks. While practical implementations and connectionless protocols can't guarantee packet delivery, the vast majority of packets are delivered correctly, and in a short time. When a packet is missing for any reason, (hopefully infrequently) the error control mechanism in NFS quietly recovers from the event and the application can proceed.

It is the exceptions that draw our attention, the situations where NFS retransmissions occur more often than expected, and users complain of degraded application performance. What can we do to minimize performance loss due to NFS retransmissions?

(1) <u>Minimize the number of devices on a given LAN</u>

By reducing the number of devices on a given LAN segment, the total load on that LAN is minimized. This reduces the burden placed on LAN controllers, host protocol implementations, and internetworking devices, while also reducing channel congestion.

(2) <u>Properly Segment Networks</u>

Co-locating users (clients) and their resources (servers) on the same LAN has a number of benefits, including: reduced internetworking traffic, improved application performance, and improved network security

(3) <u>Use Good Host Protocol Implementations</u>

With good (i.e., high-performance, well-designed) protocol implementations, the discard or loss of packets due to protocol processing and buffer limitations can be minimized.

(4) Use High Performance Bridges and Routers

While a high-performance internetworking device cannot reduce congestion caused by traffic load patterns, it can greatly reduce packet discard due to limited device capabilities. Bridges and routers that can handle data at “wire speed” can avoid unnecessary packet discard.

(5) Increase Channel Bandwidth

The more bandwidth available to the stations, the lower the likelihood that they can create short-term overload conditions. While no one should spend money wastefully, the judicious application of high-speed LAN technologies (e.g., Fast Ethernet or FDDI) on critical segments can reduce packet discard due to channel congestion.

(6) Solve the Capture Effect

The capture effect can be a major cause of frame discard when using modern, high performance hosts and Ethernet implementations. This is discussed in detail below.

3. Capture Effect

Capture effect is the term used to describe a well-known and understood idiosyncrasy of the Ethernet Medium Access Control (MAC) backoff algorithm [4]. It is considered a minor flaw in the original Ethernet design, but is now firmly entrenched through formal specifications, international standardization, and numerous silicon implementations. Before the development of modern, high-performance LAN controllers and systems were possible, the effect was rarely (if ever) seen, and did not impact higher layer protocol operation or user performance. The emergence of networked systems capable of offering continuous, high load to an Ethernet made the capture effect visible and focused attention on its impact and solutions.

The capture effect arises from the way that the Ethernet (and IEEE 802.3) specify the backoff algorithm [1] [2]. Remember that the backoff algorithm is the method by which stations reschedule their transmissions when the arbitration procedure determines that more than one station is attempting to transmit at the same time (i.e., a collision has occurred).

3.1. Design Goals

The Ethernet is designed to provide *fair access* to all stations. That is, no station has an inherent priority over any others, nor is any particular class of traffic considered privileged and granted a higher level of performance. In contrast, IEEE 802.5 (Token Ring) and 802.4 (Token Bus) systems provide a mechanism for station or traffic prioritization.

Fairness is a characteristic which must be measured over a period of time. Clearly, during any successful transmission, one station is transmitting (i.e., is being granted access to the channel) and other stations are deferring their transmissions. For that instant, the transmitting station has a “higher priority”. However, over time, all stations have an equal ability to acquire the channel and use it to transmit their data. Nothing in the algorithm provides an unfair advantage based on MAC addresses, backoff computations or anything else.

Ethernet stations must overcome a number of implementation challenges to ensure this fairness, including:

- Precise implementation of the formal state machines from the relevant specifications,
- Accurate timing of interframe gaps and backoff times, and
- Generating uniform, statistically-independent random numbers for backoff calculation.

All of this is done to ensure fair access to the network, even at the expense of reducing one’s own access and throughput.

3.2. Algorithm Assumptions

The Ethernet MAC allows stations sharing a common underlying channel to properly manage their access to that channel. It decides when a given station should transmit, and when it should defer (hold off) its transmissions to maximize total throughout, efficiency and fairness.

When offered load is light, the Ethernet provides extremely low access time for stations with traffic to send. In the simplest case, an idle channel is instantaneously available to the first station with a frame to send. As the offered load to the channel increases, the result is longer waiting times for stations wishing to transmit. This is accomplished by having stations *back off* their retransmission attempts in the face of collisions, thus allowing stations to transmit their data at the expense of others having to wait longer.

Each station must make some assumption about the current offered load to the channel, so that they can back off appropriately. If the load is extremely high, then stations should (on average) back off for a longer time. If the load is relatively light, performance can be improved by backing off (on average) for shorter periods.

Each station uses the number of times a given frame has encountered a collision as its metric for estimating the instantaneous offered load. If a frame encounters a collision upon its first attempt at transmission, the station makes an assumption that at least one additional station is offering load at this time—clearly a valid assumption. If the same frame encounters a collision on its second transmission attempt (after the back off from the first attempt), the station increases its estimate of the number of stations offering load from one to three. A collision on the third attempt increases the estimate to seven, etc. Table 1 shows how a station estimates offered load and adjusts the range of backoff times for each successive collision.

Collision on attempt #	**Estimate of # of other stations offering load ($2^{\text{attempt}}–1$)**	**Random # range ($0…2^n–1$)**	**Range of backoff times (Rand # * 51.2 µs)**
1	1	0…1	0…51.2 µs
2	3	0…3	0…153.6 µs
3	7	0…7	0…358.4 µs
4	15	0…15	0…768 µs
5	31	0…31	0…1.59 ms
6	63	0…63	0…3.23 ms
7	127	0…127	0…6.50 ms
8	255	0…255	0…13.1 ms
9	511	0…511	0…26.2 ms
10	1023	0…1023	0…52.4 ms
11	1023	0…1023	0…52.4 ms
12	1023	0…1023	0…52.4 ms
13	1023	0…1023	0…52.4 ms
14	1023	0…1023	0…52.4 ms
15	1023	0…1023	0…52.4 ms
16	too high	n/a	discard frame

Table 1: Backoff Calculations

As can be seen from the table, stations increase their backoff ranges exponentially with increasing observed offered load. This is known as *truncated binary exponential backoff*, as the exponential increase is stopped (truncated) at the tenth attempt. (This is the reason for the inherent 1024 station limitation on the extent of an Ethernet.)

3.3. The Capture Effect

The capture effect results directly from the Ethernet MAC algorithm, and imposes a short-term unfairness. To understand the effect, consider a pair of stations, each with an "infinite transmit queue", i.e., both stations always have a frame to send, and can continuously offer load to the Ethernet. (The effect occurs even if this is not the case, but it is easier to visualize it this way.)

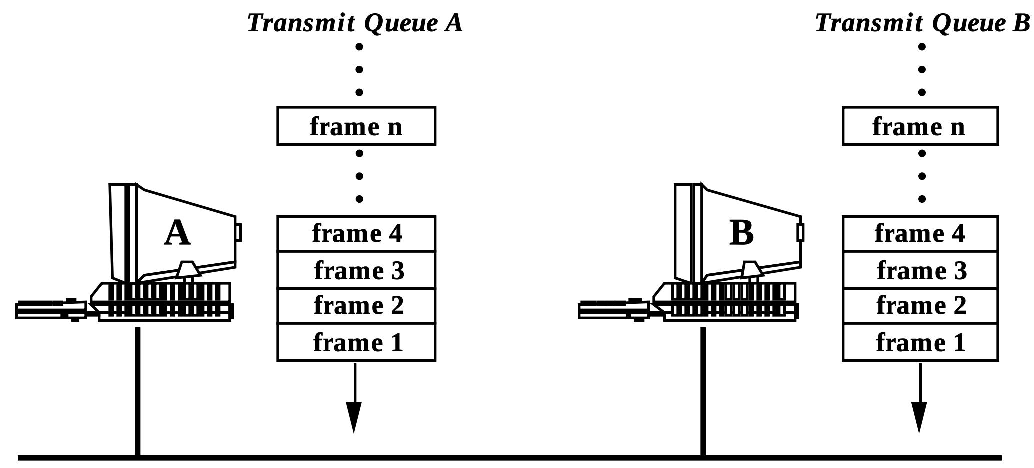

Figure 4: Capture Effect Scenario

Assume that the network is quiet at some time (either it is idle, or a frame from some other station has just completed transmission). Now our pair of stations has their queues of frames to send. We are guaranteed that both stations will experience a collision upon their first transmission attempt after seeing the channel free. (They both have a frame to send, and will arbitrate for the use of the channel.) Each will transmit, abort, and calculate a random number for backoff in the range of 0…1.

If they pick the same number (0 or 1), they will encounter another collision and repeat the process. Eventually, on some retransmission attempt they will pick different random numbers. One station (e.g., A) will acquire the channel, and the other (B) will keep backing off. Let's say that A acquired the channel by picking 0, and B lost by picking 1. A sends its frame, and successfully completes its transmission, while B waits.

However, when A has finished, both stations still have a frame to send. A will go on to the next frame in its transmit queue, and arbitrate for the use of the channel. Once again, we are assured of a collision, since B still has a frame to send. But now, it is the FIRST collision for station A (he has reset his collision counter as a result of successfully sending the first frame), but the SECOND for B.

When they pick random numbers, A will pick in the range of [0…1], and B will pick in the range of [0…3]. The probability that B will win over A is only 1 in 4 (B picks 0 and A picks 1). More likely, A will win or there will be another collision. Assuming A wins, we go through this same scenario again. But now its even worse, since it will be collision number ONE for A (again!), but number THREE for B!

In all likelihood, A will be able to transmit frames at will, and B will lose the backoff "dice toss" every time, until B gives up and discards its frame after sixteen unsuccessful attempts. Now we are back on even ground (both stations have the same collision count), and the arbitration method is once again fair.

We can see that A effectively "captures" the network for multiple successive transmissions, due to having won the first (or first few) random number contests. In fact, it is not necessary that one station (e.g., A) capture the network for B to go through 16 attempts and a frame discard. There may instead be many stations on the LAN with a combined high offered load, in lieu of a single station with an infinite transmit queue. That is, the capturing "station" may be multiple stations. It is not so much that one station "captures" the channel, but that some station (B, in our example) is locked out and ultimately discards a frame.

Over long periods of time, the Ethernet is still fair. There is an equal probability for either A and B to capture the network. During the "capture period", some station(s) has a higher probability of acquiring the channel than others, but after sixteen collisions, the network again reverts to equal access probability for the previously locked-out station.

There are two important outcomes from the capture effect:

(1) The variance in transmission delays is much greater than expected. Under high offered load, stations will tend to either transmit their frames quickly (the capturing station), or defer for very long periods (the loser, who undergoes sixteen collisions/backoffs/retransmissions, taking tens-hundreds of milliseconds before successfully sending a frame).

(2) The station not capturing the channel DISCARDS A FRAME. This is significant for higher layer protocols (such as NFS) that must recover from frame discards.

3.4. History and Evolution

When the 10 Mb/s Ethernet was being designed (1979-80), very few stations could take advantage of such a high channel data rate. In fact, Ethernet was often ridiculed early on for being faster than anyone needed, and as a result, costing more than necessary.

Early Ethernet implementations were unable to sustain back-to-back transmission attempts over extended periods due to:

- Interface hardware limitations (including early Ethernet controller chips) unable to maintain 10 Mb/s transfer rates,
- Device driver performance limitations,
- Host and higher layer protocol processing limitations, etc.

If stations cannot sustain continuous back-to-back transmission, then the capture effect cannot be maintained over sixteen retries. If a station which has “captured” the channel ever pauses in its transmission attempts, the other station is ensured of an opportunity to get a frame transmitted without a collision. This eliminates the frame discard and the need for higher layer retransmission endemic to the capture effect.

Unfortunately, newer high-performance controller and systems implementations can result in stations always having a frame queued for transmission, creating the capture effect scenario.

3.5. Symptoms of the Capture Effect

We have seen how the capture effect can cause the “loser” to discard a frame which had been queued for transmission by a higher layer protocol (e.g., IP). Most such protocols have no mechanism for re-queuing this frame automatically. They depend on even higher-layer protocols (e.g., TCP or NFS) to recover, using end-to-end error control. Frames discarded in such a manner may be observable through:

❑ Network Management Statistics

Most network management systems keep extensive statistics on important events at all layers of the architecture. Typical managed objects within an Ethernet include a count of frames discarded due to sixteen successive unsuccessful attempts (e.g., SNMP dot3StatsExcessiveCollisions, per [3]). Unless the Ethernet is severely misconfigured (e.g., the steady-state offered load far exceeds the Ethernet’s 10 Mb/s channel data rate), a high count of frames discarded due to sixteen successive collisions may indicate that the capture effect is operating.

❑ High Layer Protocol Retransmissions

Since missing frames are generally recovered through end-to-end error control in higher layer protocols, a high retransmission count (e.g., NFS retransmissions) in an otherwise stable and error-free network may indicate capture effect.

- ❑ Operating System Messages

  Some operating systems may provide an immediate operator warning message in the event of an excessive-collision-error in the Data Link (e.g., “Net Jammed”). Persistent messages of this nature in an otherwise satisfactory operating environment may indicate capture effect.

- ❑ Performance Degradation

  Ultimately, capture effect can affect user throughput and performance. If the effect persists, packets will be missing at a much higher rate than anticipated by the system designers. This will cause an increase in the time spent recovering from these events by higher layer protocols, which may measurably degrade end-to-end throughput.

  At this point, it is important to note that capture effect is not the only (nor even the most likely) cause of performance degradation in real networked systems. It is simply one possible cause. If capture effect is suspected, it can be verified through more direct observation (e.g., dot3StatsExcessiveCollisions). Users should be careful not to blame all performance problems on Ethernet capture effect.

3.6. Solutions to the Capture Effect

The capture effect results directly from the specification of the backoff algorithm in Ethernet and IEEE 802.3. Network architects have known about this idiosyncrasy for many years, and its causes and effects are well-understood. A complete “solution” to the problem will require changing the industry-standard specifications.

That is precisely what is happening. At the time of this writing (January 1995) the IEEE 802.3 committee has authorized a study group to evaluate changes to the Ethernet MAC algorithm to neutralize the capture effect. (The group is also addressing a number of issues having nothing to do with capture effect, but which affect the Ethernet MAC.)

One promising alternative algorithm, known as the *Binary Logarithmic Arbitration Method* (BLAM), has been proposed and is being given serious consideration. The details of BLAM are beyond the scope of this paper. BLAM is documented in a publicly-available technical report [4].

An alternative to changing the Ethernet specification is to modify the behavior of high-performance stations so that capture effect is less likely. Remember that the effect results from stations taking adopting the most “aggressive” behavior permitted under the specifications, i.e., continuously offering load, and using exactly the backoff time provided by the standard algorithm. If a high-performance station “tilted the scales” slightly, so that it was less aggressive than it was allowed to be under the rules, then it would not capture the channel for long periods of time.

This is perfectly permissible under the industry-standard specifications [1] [2]. There is no prohibition against taking a *less aggressive* posture regarding backoff times than permitted (i.e., backing off longer than the algorithm would require). The result will be that the network will favor access by stations using the unmodified algorithm relative to stations taking the less aggressive posture. This is exactly what we would want to do in a high-performance station concerned with minimizing capture effect.

Some Ethernet controller ICs allow for just such a modified algorithm. The AMD 79C900 (ILACC™) [5] provides a mode where the backoff counter (which measures the time waited before rescheduling a transmission following a collision) is inhibited when there is activity (CarrierSense) on the channel. This increases a station’s backoff delay during periods of heavy load, which has precisely the desired effect on the network.

The modified backoff algorithm reduces the probability of capture effect, at the expense of reducing the effective “priority” of the station implementing the modified backoff. The use of such a modified backoff algorithm is appropriate in many-to-one (server-based) environments where the server implementation is capable of “capturing” the network. By lowering the priority of the server, there will be a natural throttling of traffic by the user applications. Clients are given effective priority, but since virtually all of their traffic is directed to or from the server, the server is never prevented from accessing the network, even with its lower priority. The applications running in the client prevent the client from hogging the network to the exclusion of the server. Total performance is increased compared with the unmodified backoff which allows the server to capture the network and lock out some clients.

This solution would be less effective in a many-to-many (peer-peer) environment, since it would allow pairs of high-performance clients participating in direct peer communications to effectively prevent the server (or any device) using the modified backoff algorithm from accessing the network. As long as most traffic is server-directed, the “lowered priority” of the server is not a problem, and actually improves overall performance.

4. Collision Statistics and Network Performance

A major preoccupation with network administrators these days seems to be monitoring and worrying about the number of collisions seen on Ethernet networks. There is a great deal of folklore and voodoo concerning what is an “acceptable” collision rate or collision percentage, and when is the network “broken” or on the verge of collapse. Except in the most extreme of circumstances (all of which are observable through other, better metrics), the number of collisions seen on a network in an uninteresting and misleading statistic.

4.1. What *is* a Collision?

Perhaps the biggest mistake made by the original Ethernet designers was using the term *collision* to describe that (now well known) event. Since most collisions encountered in everyday life (i.e., collisions between trains, cars, and people) are normally to be avoided at all costs, people assume that the same is the case with Ethernet collisions. This is simply not true. A collision is just the mechanism used by Ethernet to control access to the shared medium among all users wishing to transmit data at any given time.

We can describe the operation of an Ethernet transmitter in an entirely different, but equivalent manner to that normally used:

- ❑ A station with data to transmit first allows transmissions already in progress (at the time they have the frame to transmit) to finish.

- ❑ After the completion of any frame-in-progress, a station wishing to transmit must then *arbitrate* (i.e., go through a selection system) to determine whether they have permission to transmit.

- ❑ The method of arbitration used is for the station to begin transmitting the frame that they wish to send.

- ❑ If, while transmitting the frame, the station is NOT informed otherwise, then the station has been granted permission (“won the arbitration”). The station completes the transmission and is finished.

- ❑ If more than one station is arbitrating, then they are all informed of these multiple simultaneous arbitration attempts. Being so informed, each station quickly aborts its attempted data transmission, and reschedules its attempt for a later time.

Described in this manner it is clear that a “collision” is not an event to be avoided; collisions are “good”. They are the mechanism Ethernet uses to allocate shared bandwidth among stations wishing to use the channel at the same time.

The key to efficient channel utilization in an Ethernet is two-fold:

(1) The mechanism used for channel arbitration is the same as that used to send data. There is no time wasted in arbitration if a station is granted the channel. Since this is the typical situation (one station sending at any given time), an Ethernet provides an extremely low delay (essentially zero) for typical channel access.

(2) When more than one station wishes to use the channel at the same time (i.e., a collision occurs), resolution occurs very quickly. The station almost immediately aborts the transmission, gets off the channel, and reschedules the frame. Very little channel time is wasted for the arbitration as compared to valid data transmission times.

An increase in the number of collisions on an Ethernet is therefore not necessarily indicative of a problem; it only indicates that there is more offered load. Ethernet uses the collision information to quickly redistribute the instantaneous offered load over the available time, maximizing channel utilization and application throughput.

4.2. The Effect of Increased Load on Collisions

Higher performance stations, and increased application use of the network will increase the offered load to the Ethernet. The ability of modern end-stations and LAN controller implementations to queue back-to-back frames for transmission virtually ensures an increase in the number of collisions, both in absolute terms and as a percentage of the total frames transmitted.

If two or more stations on an Ethernet have a frame queued for transmission at any given time, they are *guaranteed* by the arbitration algorithm to collide. Again, this is not *bad*, this is normal and expected. The collision will quickly resolve and both stations will be able to send their data. (The alternative is to waste time arbitrating even when only one station has data to send. This is the scenario for token-passing networks.) If new controllers and host implementations are able to queue frames more quickly than before, then the number of collisions seen on the network will increase.

Thus, the number of collisions is a measure of the load being offered to the Ethernet by the attached stations. Offered load (and therefore collision counts) will increase with:

- The number of active stations (both clients and/or servers) using the channel,
- The processing performance of those clients and servers (i.e., their ability to present load to the network), and
- The ability of new hardware and software to sustain increased performance demand.

An increase in any or all of these factors will increase the offered load to the Ethernet and result in higher measured collision statistics, both in absolute terms and as a percentage of total frames transmitted.

Collisions do not use a large percentage of available channel bandwidth, even under moderate-to-heavy offered load. Thus, an increase in observed collision counters is not a problem indication *per se*. As long as user performance and application throughput are acceptable, collision statistics can be generally ignored. An increase in the collision counters while maintaining acceptable user performance is an indication that the Ethernet access control algorithm is properly managing the available channel bandwidth.

4.3. What is Reasonable Load?

The question can then be asked, "If collision counts are not especially useful, what *is* a reasonable metric for load, and what constitutes an acceptable load for an Ethernet?". A complete treatment of this question is beyond the scope of this paper, but some important observations and rules-of-thumb can be made.

First, it is much more useful to measure channel utilization (percent of time that the channel is busy, or carrying data) than to count collisions. This is a much more direct measurement of the load on the channel.

Second, it is meaningless to discuss utilization without noting the time period over which the utilization is being averaged. For example, the statement: "The network utilization is 30%." is meaningless. Ethernets (and most LANs) work by time-division multiplexing the shared channel among multiple users. At any given time, the channel is either in use (100% utilized) or not in use (0% utilized). It cannot be "30% used". So we must always note the time period over which the utilization is averaged. 30% utilization over a 1 minute period can be reasonably expected under many conditions; 30% utilization over a 24 hour period would imply a high offered load.

There are many variables to consider when trying to determine what constitutes an acceptable utilization:

- Number of stations on the LAN
- Application behavior
- Traffic patterns
- Frame length distribution, etc.

Nonetheless, experience shows that for many common environments, including office automation LANs with tens of stations, the following utilization levels can be used as "rules of thumb" for determining when a LAN is approaching excessive load:

❑ Utilization exceeds 10-20% averaged over an 8-hour work day, or

❑ Utilization exceeds 20-30% averaged over the worst hour of the day, or

❑ Utilization exceeds 50% averaged over the worst 15 minutes of the day

Note that for very short-term periods (seconds, or even tens of seconds), network utilization may be nearly 100% without causing any problems. This might occur during a large file transfer between a pair of high-performance stations on an otherwise quiet network.

Again, these are not hard-and-fast rules, and some application environments may operate well under heavier loads or fail at lighter levels.

5. Summary

The widespread use of high performance workstations and servers using more capable Ethernet controllers can result in unexpected behaviors in systems and applications. It is important to understand the source of these behaviors in order to determine whether they constitute true problems, or are simply artifacts of the new implementations.

This paper considered the problems of:

❑ NFS retransmissions and performance degradation due to missing packets, including possible sources of packet loss and discard,

❑ The Ethernet capture effect, and

❑ Increased collision rates.

While these are not the only issues to be considered, they are all significant and observable in practical network environments.

The author wishes to thank: John Cassidento, Paul Chan, Brian Conley, Mike Falco, Clive Philbrick, Vern Schryver, and Bob Stewart for their technical input, lab support and document review, and Casey Cox and Steve Wierenga for logistical support.

6. References

[1] ISO/IEC International Standard 8802-3, Information technology-Local and metropolitan area networks, *Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*, Fourth edition 1993,

[2] Digital Equipment Corp., Intel Corp., Xerox Corp., *The Ethernet: A Local Area Network, Data Link Layer and Physical Layer Specifications*, Version 2.0, November 1982

[3] F. Kastenholz, RFC 1643: *Definitions of Managed Objects for the Ethernet-like Interface Types*, IETF, July 1994

[4] Molle, Mart L., *A New Binary Logarithmic Arbitration Method for Ethernet*, Computer Systems Research Institute, University of Toronto, Technical Report CSRI-298, available by anonymous ftp: cs.toronto.edu/reports/csri/298

[5] Advanced Micro Devices, Ethernet/IEEE 802.3 Family Data Book, ©1992

Networks and Communications Consulting specializes in providing consulting services to manufacturers, systems integrators, and users of computers and computer communications systems. Our staff has extensive experience in network systems architecture, technical marketing**,** network hardware and software implementation, enterprise-wide network design, network standards, and communications systems design. We have been providing quality marketing, training, hardware, and software services to a wide range of clients, internationally, for more than 7 years.

RICH SEIFERT, M.S.E.E., M.B.A., is President of *Networks and Communications Consulting*, Cupertino, CA. Formerly with Digital Equipment Corp. and Industrial Networking, Inc., he was responsible for the development of the Ethernet physical layer and specifications, and Token Bus Factory LAN products. He is co-author of the IEEE 802.1, 802.3, 802.4 and Fast Ethernet standards, and is currently working on Wireless LANs, Internetworking, Protocol design, High Speed networks and new network architectures. He teaches courses on networking for the University of California at Berkeley and many private companies.