The Ethernet Capture Effect: Analysis and Solution

K. K. Ramakrishnan and Henry Yang
Distributed Systems Architecture and Performance
Digital Equipment Corporation
550 King Street, Littleton, MA.
rama,yang@erlang.enet.dec.com

Abstract

We analyze the behavior of the Ethernet in networks with a small number of active stations, and describe what is known as the Ethernet capture effect, where a station transmits consecutive packets exclusively for a prolonged period despite other stations contending for access. The capture effect causes transient unfairness, which results in substantial performance degradation. We report measurements using TCP/IP that show the performance degradation. A solution is proposed that effectively overcomes the capture effect. The proposed algorithm, which we call the Capture Avoidance Binary Exponential Backoff (CABEB), uses the standard Binary Exponential Backoff (BEB) with enhancements for collision resolution in the special case when a station attempts to capture the channel subsequent to an uninterrupted consecutive transmit. Using a detailed simulation, we show the efficacy of the CABEB algorithm over the standard BEB in overcoming the unfairness resulting from stations capturing the channel. The CABEB improves throughput for protocols like TCP/IP, reduces variability in the channel access latency and eliminates packet discards due to excessive collisions in a 2-node network. The algorithm is a modification that is compliant with the Ethernet/802.3 standards. For networks with a large number of active stations, the CABEB performs as well as the standard BEB algorithm. Our study places emphasis on the workload and network configuration that is the worst case relative to the Ethernet capture effect to show that the proposed algorithm is a substantial improvement over the existing backoff algorithm.

1. Introduction

Ethernets have been in widespread use for over a decade now [8]. It has been the core technology enabling distributed computing, and networking for the desktop, server and backbone over the years. It has met the performance needs of applications and systems, and has been a robust and inexpensive local area network so that computer systems have begun to have an interface to it as a default [2]. The algorithms for controlling access to the Ethernet network have been examined by a large number of researchers over the years (e.g., [7,13]), and although improvements have been suggested for niche applications [6], there has been no significant change in these algorithms for the last decade. The analysis and solution proposed in this paper is equally applicable to the 10 Mbit/second Ethernet/802.3 networks and the 100 Mbit/second Fast Ethernet standard being proposed in the IEEE 802.3u Committee [4].

We now have begun to see the proliferation of smaller work group and point-point Ethernet links between servers or end-systems and a switching node, that allows for the migration from a shared network to networks where nodes are primarily connected through switches or other interconnecting devices (routers, bridges, repeaters). This has resulted in increased use of Ethernets as point-point link with 2 nodes and connecting small work groups. With this subtly altered usage pattern of Ethernets, we choose to once again re-examine the efficacy of the algorithms for channel access. This is needed also because computer systems connected to these networks have become substantially faster and can transmit and receive packets back-to-back on the Ethernet with ease. The performance degradation suffered by these systems due to the well-known *Ethernet Capture Effect* is significant for a network with small number of nodes, especially for a 2 node network.

The Ethernet Capture Effect is the behavior wherein under high load, one station is able to hold on to the channel to transmit packets consecutively, in spite of other station(s) contending for access. This is particularly acute in the case of a 2-node network, with one station receiving an unfair share of the channel bandwidth over a transient period. The number of packets consecutively transmitted by the node capturing the channel can potentially be hundreds of packets or more, if the station has this large number of appropriate size packets to transmit. The capture effect is due to the *transient unfairness* of the standard Binary Exponential Backoff (BEB) used in the Ethernet/802.3 network [1], for collision resolution (we use the terms Ethernet, 802.3 and CSMA/CD networks interchangeably). The degree of transient unfairness is severe for small numbers of contending stations and it reduces quickly as the number of active stations increases (we use the words node and station interchangeably). It also has severe performance repercussions even with protocols using window flow control with a modest maximum window size of 32K or 64K bytes, such as TCP/IP. The capture effect results in the channel being unnecessarily idle, thus reducing the overall throughput achieved by applications. The *transient unfairness* also results in the access latency seen by a station to have substantially increased variability, which is bad for emerging applications such as multimedia networking.

We study in this paper the impact of the Ethernet capture effect on the performance of popular protocols such as TCP/IP [9,10] and UDP/IP [11] in networks with a small number of nodes, and suggest that there is a clear need to consider modifications to overcome such performance degradation. The conclusions from the analysis are equally applicable to networks with a large number of nodes, but where a small number of active nodes are the source of bursty traffic for a given period of time. We then provide a solution to the capture effect problem of the BEB, which we term the *Capture Avoidance Binary Exponential Backoff* algorithm (CABEB), which we believe has all the properties needed to overcome the performance degradation without violating in any way the letter or spirit of the Ethernet/802.3 standards for access to the network. The CABEB is an extension of the standard BEB, where it uses the standard BEB algorithm for all cases of collision resolution except for one. The CABEB for collision resolution minimizes the occurrence of capture by a station by using an enhanced backoff algorithm when a collision occurs on the second packet of what is termed as an *uninterrupted consecutive transmit* by a station. This is when a station is able to transmit a second packet without another station transmitting an intervening packet.

We show that the CABEB algorithm is significantly superior to the standard BEB algorithm in the 2-node environment for the various ways that the channel may be used (e.g., TCP/IP-like window flow-controlled interaction; UDP/IP-like uncontrolled transmission of an unlimited number of packets). We emphasize in our study the worst case environments, both in workload and network configuration, relative to the Ethernet capture effect. We also show that the behavior of the new CABEB algorithm is at least as good as the standard BEB algorithm when we increase the number of stations from 2 to a reasonably large number (13 as reported in this paper). The CABEB algorithm has been incorporated without much difficulty in some of Digital Equipment Corporation's newer Ethernet interfaces and chips [15, 16].

In the next section we provide motivation for studying the problem through the analysis of a measurement experiment with 2 nodes in the network and outline the consequences of the Ethernet capture effect on emerging networking applications that depend on the availability of reasonable real-time characteristics for the network. Section 3 provides a description of the performance metrics we use as the basis to compare different algorithms for efficacy in using the network. We then describe the Ethernet Capture Effect in detail. We study the algorithms using a simulation, which we describe briefly in Section 5. Section 6 presents the results of a simulation for 2-node as well as multiple nodes for both the standard BEB and our proposed CABEB algorithm. Section 7 summarizes the work.

2. Problem Motivation

We first present some early measurement experiments performed with Alpha AXP™ systems with Ethernet interfaces that are compliant with the Ethernet/802.3 standards for access to the channel. This provides some understanding of the extent of the problem observed when a small number of actual systems use the Ethernet, when they are capable of generating significant load on the channel to expose the deficiencies of the standard channel access algorithms.

2.1 Measurement Experiments Exhibiting the Ethernet Capture Effect

We conducted several measurements of applications using TCP/IP and UDP/IP to communicate between two systems. This was performed with pairs of different Alpha AXP systems running DEC OSF/1™. When measurements were performed between two systems using the AMD LANCE Ethernet interface [17], we saw that the overall throughput with TCP/IP was markedly higher than with two Alpha AXP systems using an Ethernet interface chip (produced by Digital Equipment Corporation, called the Third Generation Ethernet Controller (TGEC) [14]) that is compliant with the behavior allowed by the IEEE 802.3 standard.

We used a tool called 'inett' (which is similar to ttcp) for benchmarking 2-node running TCP/IP or UDP/IP to measure the throughput at the application level. Inett basically transfers a specified number of messages of a given size, from the user's application space, using UDP/IP or TCP/IP. It also allows for the specification of the size of the socket buffer on the transmitter and receiver. The application transfers the data, maintains timing and reports various performance measures at the end of the experiment. We used this as the load generator between two peer systems. We used a SNIFFER™ to monitor data transfer over the Ethernet and also collected 'netstat' data at the sender and receiver before and after the experiment [5].

Observing the throughput with TCP/IP, there were significant differences in behavior between the measurements with the DEC Ethernet Interface chip (TGEC) and those with the LANCE. The overall throughput between Alpha AXP systems with TGEC with TCP/IP was 6.82 Mbits/second with a reported CPU utilization of 11 to 12% (both transmitter and receiver). The overall throughput between 2 Alpha AXP systems with the LANCE with TCP/IP was 8.65 Mbits/second, with a reported CPU utilization of about 13%. With the TGEC, we also observed a higher rate of collisions.

When sending 1000 packets, with TCP/IP (with 32 KByte socket buffer, 1460 byte user data) we observed with the TGEC systems that at the sender there were overall 530 collisions (386 single and 144 multiple collisions). On the receiver, we observed 249 collisions (154 of these experienced multiple collisions). On the other hand, with the LANCE systems, for the same experiment, there were *no* collisions.

We first measured the raw capability of the systems to transmit packets over the Ethernet, to ensure that there was no inherent limitation with either the system or the Ethernet chip in transmitting at the full rate. For this, we report the measurement with each of the systems transmitting UDP packets as fast as possible. The peak transmit rate with UDP, while sending 1460 byte user messages was 9.68 Mbits/second with the TGEC based system. We measured this by sending 10,000 messages to ensure adequate statistical accuracy. We repeated the identical experiment with a pair of LANCE based systems. The peak transmit rate was comparable, 9.65 Mbits/second. From these measurements, at a very high level, we concluded that the TGEC system and the LANCE systems are both capable of transmitting close to the line rate on Ethernet.

At this point, looking at the actual packet sequence with a SNIFFER Ethernet analyzer, we observed several interesting characteristics with the TGEC systems. We saw many occurrences of a long sequence of data packets going from source to destination before an almost corresponding long sequence of acknowledgment packets in the reverse direction. For example, with a 32 KByte socket buffer at either end, we saw several occurrences of a full window's worth of data packets flowing in one direction and then a set of acks. flowing back in the other direction.

A trace of 1000 data packets with TCP/IP and their associated acknowledgments was collected. We focused on the time from the last data packet in the stream to the time that the next acknowledgment was observed on the wire. It showed a wide distribution, ranging from 0.1 millisecond to a maximum of 46 milliseconds. We show below the actual time that the wire was observed to be idle after each value of K (the number of consecutive data packets on the wire). Table 2 is only for values of K of 20, 21 and 22. There were a few times of significance even when the value of K was smaller, but what is shown is sufficient to illustrate the behavior.

K: Number of data pkts. between acks.	**Idle Time before Ack., individual occurrences (milliseconds)**
20	8.6; 13.1; 12.8; 17.4; 15.5; 1.3
21	14.6; 46.0; 18.2; 11.5; 11.4; 14.7; 10.2; 23.8; 45.8
22	3.7

Table 2.1: Number of consecutive data pkts. send and Idle times observed on the channel between last data packet and acknowledgment sent by TCP/IP receiver.

During the time to send 1000 TCP/IP data packets, the total idle time during just the 16 occurrences of idle time on the channel listed above (which is the contribution due to the receiver staying in backoff after the transmitter's window has closed and the ack. is still awaiting to be transmitted) adds up to 268.6 milliseconds, (about 223 maximum size packets time on the Ethernet). In fact, from the reported *inett* time, the experiment took about 1.75 seconds total, including time for exchanging control packets, setting up the connections etc. The idle time for these 16 occurrences (shown in Table 2.1) was almost 15.2% of the total time, possibly explaining the loss in throughput. If we just removed this idle time the throughput would be almost 8 Mbit/second, instead of 6.5 Mbits/second Of course, we did not factor in the idle time observed in all the other cases. We believe the 'capture effect' explains a substantial part of the loss in throughput.

We then experimented with a pair of DEC Alpha AXP Workstations with the same version of OSF/1 with the LANCE Ethernet interface, to see if this capture effect is exhibited. We did not see such an extreme case.

We attributed the loss in throughput with TCP/IP to the 'Ethernet Capture Effect' in which one station on the Ethernet unfairly captures the channel to transmit a large sequence of packets, while the other station is experiencing a long collision backoff delay due to consecutive collisions on the same packet. Only at the end of the transmission of the large sequence (often equal to the complete window size of 21-22 packets), is the receiver is able to send the acknowledgments back. Often the acks. are still in their, significantly large, backoff interval. This results in idle times on the channel.

The behavior is observed with the TGEC because of its strict conformance to the Ethernet/802.3 specification for IPG after a transmission of a packet on the channel. With the LANCE, we do not see such a severe behavior. This is attributable to the inability of the LANCE to contend for the channel within the minimum IPG interval, which is a behavior allowed by the Ethernet/802.3 standard.

3. Performance Metrics

There are several performance metrics we use to examine the efficacy of the different Ethernet access algorithms studied in this paper. When considering a solution for the capture effect, we attempt to compare our solution in all of the typical dimensions - Throughput, Response Time, Access Latency, Percentage of Collisions, Percentage of Packets Discarded etc. In addition, we believe it is important to consider the fairness in allocation of the channel carefully.

- Fairness: Fairness in providing access to the channel to the different stations that are contending to transmit is an important issue. Since we are considering a single resource being accessed by multiple stations that are placing an infinite demand on it, the allocation is fair when all the contending stations get an equal share. We evaluate whether an algorithm is fair or not by looking at all of the metrics outlined above for each of the different stations on the network. We break up the consideration of Fairness into two sub-classes:

  — Long-term Fairness: When the average, variance and other measures for throughput, response time, access latency and other metrics are equal for stations placing an identical demand (offered load) on the channel, over the long term, then we consider the algorithm to have *long-term fairness*.

  — Transient Fairness: Even though the long-term average behavior of an algorithm may be fair in providing equal allocations to all the stations placing a demand on the channel, there may be *transient unfairness* in the allocations provided to stations over shorter time windows. We consider an algorithm to have *transient fairness* when the metrics such as throughput, response time, access latency and other metrics are equal for stations placing an identical demand on the channel over an arbitrarily small time window that spans at least *n* packet transmission times, where *n* is the number of stations contending for access to the channel. For example, when two stations that have identical size packets to send are contending for access to the channel, then an algorithm that provides alternating access to the stations to transmit is considered to have *transient fairness*. In these terms, when an algorithm allows a station to lock out the other station from transmitting packets for multiple opportunities, it is considered to have *transient unfairness*. A good measure of the *transient unfairness* of the algorithm is the number of consecutive transmits by a station even when another station is contending for access.

4. Problem Description: The Ethernet Capture Effect

Briefly, the ‘capture effect’ is as follows: Consider two stations on the Ethernet, each with a significant amount of data to transmit and able to achieve the minimum IPG rules. Let us say station 1 (with *data1*) and station 2 (with *ack1*) both attempt to transmit simultaneously (within a slot time of 51.2 microseconds). Each station has a collision counter, *n*, which is zero to start with. They experience a collision, incrementing *n* to 1. Each station picks a backoff time value which is uniformly distributed from 0 to (2^n-1) slots. This is now 0 or 1. If station 2 picks a backoff value of 1 (50% probability), and station 1 picks a backoff of 0, then subsequently station 1 successfully transmits its packet - *data1*. Station 2 waits for completion of *data1* before attempting again to transmit *ack1*. The collision counter at station 2 remains at 1 while the collision counter at station 1 is reset to 0. If station 1 has another packet to send, *data2*, this will now contend for the channel with *ack1*. If these collide, the backoff values chosen are: for station 1: 0 or 1 slots. For station 2: 0, 1, 2 or 3 slots since the collision counter is 2 at this station. So, there is a higher likelihood for station 1 to succeed when resolving this collision and transmit *data2*, while *ack1* from station 2 will begin deferral when it completes it’s backoff interval. We compute below, the probabilities for this behavior and show that with high probability one of the stations in the 2-node Ethernet can ‘capture’ the channel for an unfair amount of time. This would happen till a maximum number of collisions, 16, is encountered for a packet (station 2’s *ack*) at which point the packet is discarded. Station 2 now starts again with a collision counter value of 0 and potential long-term fairness may be achieved when station 2 attempts to resend *ack1*. Note that if station 1 completes transmitting a stream of packets during such a capture epoch, and station 2 is still in backoff, the channel is idle for this period of time.

In the case of a TCP flow, the capture effect results in the sender waiting for the acknowledgment after having transmitted all of packets in the window. This results in a larger overall elapsed time to complete the transfer of a certain amount of data.

For a two node network, the contention between the two nodes results in one winning to transmit a packet. After transmitting the first packet, the node has a progressively higher probability of winning future collision attempts for a back to back transmit case. The following shows the probability of winning a collision after a node (Node A) has previously won the collision between the two nodes.

Probability (Node A winning) = 1 - (3 * $2^{-(n+1)}$) **(1)**

Probability (Node B winning) = $2^{-(n+1)}$ **(2)**

where n = # collision attempts

= (2,...,9) for collision attempts 2 to 9

= 10 for collision attempts 10-15

Under heavy load, a two node network exhibits the worst case for the capture effect, where one node captures the channel.

# collision	Prob(A win)	Prob(B win)
2	0.625	0.125
3	0.8125	0.0625
4	0.90625	0.0315
5	0.95313	0.0156
6	0.97656	0.00781
7	0.98828	0.00391
8	0.99414	0.00195
9	0.99707	0.00098
10 - 15	0.99707	0.00098

Table 4.1: Two Node Network - probability of collision resolution

As shown in Table 4.1, after Node A has won a collision the probability of Node A continuing to win any subsequent collisions quickly approaches 1. Node A is more likely to be able to transmit packets back to back and win most of the collisions with Node B. During this epoch of Node A's capture, Node B will spend most of its time in the retransmission backoff state. Given its low probability of winning any collision during this epoch, Node B will eventually suffer 16 collisions before it aborts the transmission of the packet. After aborting the packet, Node B will attempt to transmit a new packet with equal probability of winning a collision, as Node A. A new epoch of capture begins with the winner of the next collision. This sequence is repeated with a winner starting each new epoch of capture. Although this results in significant transient unfairness, the long-term throughput achieved by the two stations may in fact show no long-term unfairness. For implementations with good random number generators for the backoff algorithm, the two nodes should have close to equal number of chances of capturing the channel when observed over a reasonably long period.

4.1 The Solution: The Capture Avoidance BEB Algorithm

We will briefly describe the standard BEB algorithm. When transmitting a packet, a station uses the standard BEB algorithm for collision resolution, when it encounters a collision. After detecting a collision, the station retries the transmission by backing off a random number of slot times before attempting to transmit the packet. The scheduling of the retransmissions is determined by a controlled randomization process called "truncated binary exponential backoff". A slot time is 512 bit time (e.g., 51.2 µseconds for 10 Mbps system and 5.12 µseconds for 100 Mbps system). The number of slot times to delay before the **n**th retransmission attempt is chosen as a uniformly distributed random integer **r** in the range $0 <= \mathbf{r} < 2^{\mathbf{k}}$, where **k = min (n,10)**. As the number of collision attempts increase for the same packet, the station delays its retransmission based on the truncated binary exponential function. The retransmission is aborted after 16 collisions on any given packet, and this is called an *excessive collision error.*

The CABEB executes the standard BEB for all the collision cases except for a special case. If a station transmits a first packet and begins to transmit a second packet and the station has not received a collision or a fragment or a packet between the first packet and the beginning of the second packet, we call this an *uninterrupted consecutive transmit*. The CABEB processes this uninterrupted consecutive transmit using an enhanced BEB algorithm to solve the capture effect problem. The CABEB is compliant with the Ethernet/802.3 standard.

There are several back to back transmit cases of interest. The first case is the uninterrupted consecutive transmit, where a station transmits a second packet after it has successfully transmitted a first packet and there were no other stations contending for the channel prior to the beginning of the second packet. In the uninterrupted consecutive transmit case, the second packet could encounter a collision. So, the conditions required for this case are a station successfully transmitting a first packet, the channel is idle (i.e., no packet is received, or fragment or collision encountered) for an arbitrary period of time after the first packet, and the station begins to transmit a second packet (i.e., transmit the first bit of the preamble).

The second case of interest is where a station transmits a first packet, then it receives one or more packets or fragments or collisions, and then the station transmits a second packet. We will call this case an i*nterrupted consecutive transmit*. For an interrupted consecutive transmit, the station of interest (i.e., the one that just transmitted a packet) is not involved in the collision.

The third case of interest here is when the second packet of an uninterrupted consecutive transmit is involved in a collision and the station wins the collision resolution to complete the transmission of the second packet. An uninterrupted consecutive transmit is the first portion of a captured transmit, where the second packet of the uninterrupted consecutive transmit is involved in a collision. We will call this case a *captured transmit*. When a station does consecutive captured transmits for multiple packets, we say that the stations has captured the channel. A captured transmit occurs when the station winning the collision resolution transmits the second packet. Figure 4.1 illustrates these cases.

The CABEB solves the capture effect problem by minimizing the occurrence of captured transmits. This is achieved by using an enhanced backoff algorithm when a collision occurs on the second packet of an uninterrupted consecutive transmit. For an interrupted consecutive transmit or a transmit after an initialization, it uses the standard BEB for collision resolution. It is intuitive that *at any given time on a CSMA/CD LAN, there can be no more than one station in an uninterrupted consecutive transmit state*. This forms the basis for developing our solution. For a CSMA/CD system with reliable collision detection design only one station can have a successful packet transmit (i.e., one with no collision) at any given time. So, after a station has successfully transmitted a packet and the channel is idle, this station is the only station in the LAN that can initiate an uninterrupted consecutive transmit. Given that an uninterrupted consecutive transmit is the first portion of a captured transmit, the enhanced BEB algorithm tries to avoid this captured transmit. For an interrupted consecutive transmit or transmission of a packet after initialization, the CABEB uses the standard BEB. The enhanced backoff algorithm can be described as follows. When transmitting a second packet of an uninterrupted consecutive transmit, the station takes a backoff of 2 slot times on the first collision. If the other colliding station is transmitting a fresh packet (i.e., one that has not experienced any collision), that station will draw a backoff of 0 or 1 slot time (according to the standard BEB) and hence its packet is guaranteed to be transmitted.

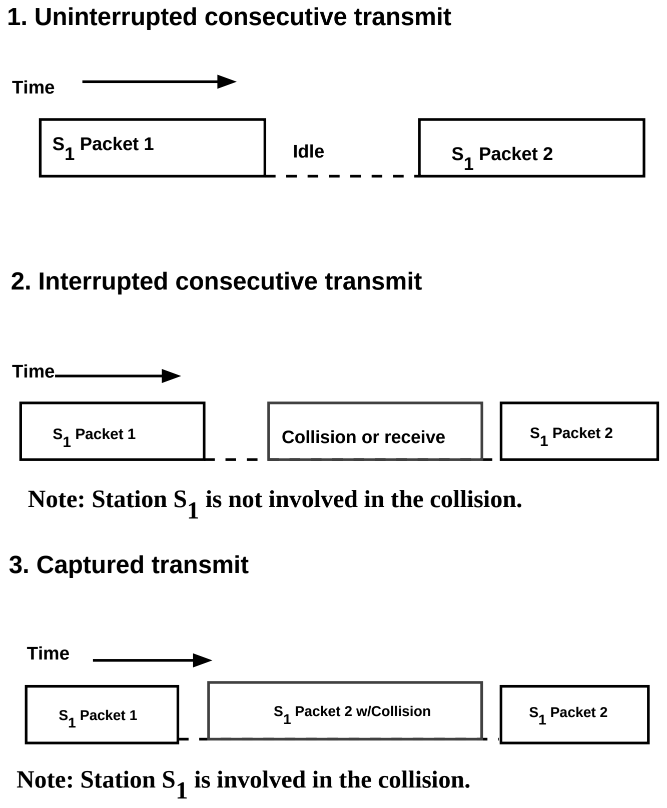
1. Uninterrupted consecutive transmit
Time
S1 Packet 1
Idle
S1 Packet 2
2. Interrupted consecutive transmit
Time
S1 Packet 1
Collision or receive
S1 Packet 2
Note: Station S1 is not involved in the collision.
3. Captured transmit
Time
S1 Packet 1
S1 Packet 2 w/Collision
S1 Packet 2
Note: Station S1 is involved in the collision.

Figure 4.1: Examples of Transmit Cases

```
CABEB Algorithm
  n = collision attempts (0-15)
  r = standard BEB uniformly distributed random number
  k = BEB backoff range variable
  backoff = number of slot times to backoff
For collision resolution, the following procedure determines the backoff time.
 For an uninterrupted consecutive transmission,
    If n=1 then backoff = 2;
    If n=2 then backoff = 0;
    If n > 2 then backoff = r;
          where $0 <= r < 2^k$, $k = \min(n, 10)$
 For an interrupted consecutive transmission,
    backoff = r;  where $0 <= r < 2^k$, $k = \min(n, 10)$
```

Thus, the CABEB guarantees that a capture transmit does not occur when both of the colliding packets are experiencing their first collision. After the backoff of 2 slot times, the station is ready to retransmit the packet. If the packet collides for a second time, the station draws a backoff of 0 slot times. Thus, the station will retransmit immediately after the IPG. The selection of 0 slot time on a second collision (of an uninterrupted consecutive transmit) allows the station to have a higher probability of winning the second collision. If the same packet experiences a third or subsequent collision, the station uses the standard BEB for collision resolution from the 3rd to the 15th collision. The following describes the algorithm more precisely.

The CABEB is not meant to be effective when one of the packets involved in the collision has advanced beyond its second collision attempt. In this case, the CABEB's backoff of 2 slot times does not help because the other station may be backing off a random number of slot times based on the collision attempt and the standard BEB. For those cases where a collided packet has advanced beyond its second collision attempt, the CABEB behavior is similar to that of the standard BEB. This is important as we would like the performance of the CABEB to be no worse than the standard BEB algorithm when the number of stations in the network is large.

For minimum size packets, the algorithm allows up to two packet transmissions for every collision as the two stations alternate their transmits. So, for minimum size packets with infinite load case, the collision rate is 50%. For maximum size packets, the algorithm only allow 1 packet transmission for every collision. In this case, the collision rate is 100%. The CABEB solves the capture effect problem at the slight expense of collision overhead. It provides significant improvement in fairness, packet discard rate and access latency.

The CABEB enhanced the standard BEB algorithm by modifying the early stages of collision resolution (i.e., the first and second collision of a packet). As a result, the enhanced algorithm is most effective for a network with small number of stations, where the enhanced algorithm avoids a given packet from advancing its number of collisions beyond two. This is guaranteed for a two node case, as the algorithm allows the two stations to alternate their transmits. In addition, the CABEB maintains the same mean as the standard BEB for the retransmission backoff time for multiple collisions on any given packet. This allows the algorithm to be compliant to the IEEE 802.3 standard. Another added advantage of maintaining the same mean value for multiple collisions is that the behavior of CABEB converges to the standard BEB for network with larger number of stations, at high load when all of the stations are active.

As will be shown later, the CABEB completely solves the capture effect problem for networks with small numbers of stations. For networks with a large number of stations, the CABEB can still be useful when the number of active stations involved in a burst of transmits is small.

5. Simulation Methodology

We used a CSIM [12] based simulation to examine the performance of the proposed CABEB and standard BEB algorithms. Details of the station awaiting the channel to be idle before transmission of a packet, waiting for an IPG interval before beginning to transmit were all modeled. The collision window was also modeled so that any two stations beginning to transmit within that interval relative to each other experience a collision. All of the station characteristics in terms of collision counter management, using it to go through the backoff interval were also modeled. The physical extent of the Ethernet was allowed to vary, as well as the kind of workload generated by the stations.

We looked at two different types of workload generated by the stations. One was to emulate the behavior of UDP/IP or a similar connectionless transport protocol, where the station just transmits as fast as allowed on the channel, and no acknowledgments are generated by the receiver. For this case, we model a varying number of stations on the network.

The other type of workload modeled emulates the behavior of TCP/IP. We create pairs of stations that are the source and sink of a TCP data stream. The acknowledgments generated by the receiver is controlled so that 1 ack. is generated every 2 data packets.

We have simulated effectively the worst case environment for studying the capture effect. We consider the maximum allowable extent for the Ethernet, which results in the highest penalty in throughput when a collision occurs. We also consider the stations to be transmitting either minimum size packets or maximum size packets, which also is looking at the boundary cases for understanding the behavior of the algorithms. We have also considered the stations to have an infinite load (interarrival time of 67 µseconds so that the simulation is at least stable) and essentially zero service time to send or receive packets so that even one station can saturate the channel. This is once again the worst case load for examining the behavior of the backoff algorithms, when the effect of capturing the channel is at its extreme.

6. Results

We show that the CABEB algorithm provides substantial improvement in the performance of TCP-like protocols without much degradation in throughput for UDP-like protocols. We present results here where the load on the Ethernet is such that it is saturated, with even one station presenting enough load to fully utilize the channel. We only present results for stations generating different fixed packet sizes, even though the simulation is easily capable of having distributions for the packet size. The motivation is to help in understanding the results, and provide almost all of the intuition needed for designing the collision resolution algorithm described here.

One performance criterion of importance is that of fairness in access to the channel. We evaluate fairness in the access by observing the mean throughput and the mean and variance in the access latency to the channel. This is the time from when the packet is at the head of the queue of packets to be transmitted at the station to when it is successfully transmitted, including the time to transmit the entire packet. A more important and dramatic metric that exhibits the *transient unfairness* in access to the channel is the number of packet discards and the number of consecutive transmits for a station.

6.1 Performance of UDP-like datagram protocol with CABEB algorithm

We first looked at the performance of a two station Ethernet network, with both stations having an infinite amount of data to transmit and using a UDP-like datagram protocol. The stations transmit fixed length packets, as fast as the channel access algorithms will allow. If a packet was discarded because of excessive collisions, then the packet is not retransmitted. Further, when a station needs to transmit, it is not limited by any flow control mechanism, and does not await an acknowledgment from the receiver before transmission of the packet. We chose to examine the behavior for an Ethernet with the maximum physical extent (51.2 μseconds round trip propagation time) with the stations transmitting 64 or 1500 byte packets (including all headers). There was no host processing time modeled, and as such only the channel access times and IPG, preamble and packet times were simulated. We ran the UDP-like simulations for 30 seconds in the 2-node case and for 5 seconds for the results relating to multiple nodes.

Table 6.1 shows the performance of a 2-station network using such a datagram protocol, and using either the standard Binary Exponential Backoff (BEB) algorithm as defined in 802.3 ("*Old*") or the CABEB algorithm ("*New*"). With both the stations being *old*, we find that the throughput on the Ethernet reaches over 7.56 Mbits/second with 64 byte packets. The percentage of packets experiencing one or more collisions encountered by each of the stations is in the range of 0.55%. While the collision statistics are quite good with the *old* algorithm, we found that station 1 discarded 79 and station 2 discarded 83 packets each due to excessive collisions, while transmitting a total of 443465 packets in the 30 seconds of simulation. This meant a discard rate of 0.035% to 0.038% for each station. This is a significant packet loss rate, especially for datagram transport protocols, since the application has to potentially retransmit the complete message. Another observation we make is that during each of these periods when a station is going through the 16 collisions, it is not transmitting any packets on the channel, and the other station transmits several hundred of packets consecutively. The station losing the collision resolution experiences a prolonged period of unfairness. Even observing the average throughput achieved by the two stations, we see that station 1 gets a throughput of 3.851 Mbits/second, while station 2 only achieves 3.717 Mbits/second, indicating that the capture effect in fact results in unfairness that is not just a transient one.

Examining the case with the two stations being *new*, we observe that the throughput on the Ethernet reduces significantly, down to 5.424 Mbits/second. This is due to an increase in the number of collisions experienced by the two stations. The deterministic nature of the CABEB backoff algorithm results in every other packet for a station encountering a collision. We observe from the table that there are no packets discarded with the CABEB algorithm. Furthermore, the throughput on the Ethernet is divided in a completely fair way between the 2 stations. In fact, observing the time sequence of events in accessing the channel, we found that the stations alternated in access to the channel. The statistics for the access times, both mean and the variance, for the two stations are identical. Therefore, the CABEB algorithm eliminates the capture of the channel by a station in the 2-node network.

We also examined the performance with a mixture of station 1 being a *new* station and station 2 being an *old* station. We find that the overall throughput degradation due to the *new* station is not significant, achieving 7.542 Mbits/second compared to 7.568 Mbits/second with the *old* stations. There is however, even more unfairness, with the *new* station receiving a greater share of the Ethernet bandwidth. There are packet discards, since repeated collisions are no longer eliminated with the introduction of the *old* station. But, the *new* station experiences fewer packet discards than the *old* station (0.032% vs. 0.042%).

We then consider the performance comparison between the CABEB algorithm versus the standard BEB algorithm for the stations transmitting 1500 byte packets. With the standard BEB algorithm, the overall throughput achieved on the

Ethernet is 9.806 Mbits/second, while having a total of 151 packets discarded (out of 24516 packets transmitted). The percentage of packet loss due to excessive collisions has now gone up to 0.53% (station 1) and 0.71% (station 2), which is excessive. The unfairness due to the standard BEB *old* algorithm is also significant, with station 1 getting 5.32 Mbits/second compared to only 4.486 Mbits/second for station 2.

Types of stations	**pkt size (bytes)**	**Enet tput (Mbps)**	**Station 1**	**Station 2**
			Thruput (Mbps)	Thruput (Mbps)
New-New	64	5.424	2.712	2.712
Old-Old	64	7.568	3.851	3.717
New-Old	64	7.542	4.059	3.483
New-New	1500	9.446	4.723	4.723
Old-Old	1500	9.806	5.32	4.486
New-Old	1500	9.771	5.379	4.392

Table 6.1: UDP Performance with two New (CABEB) stations and/or Old (standard)stations (30 second simulation)

Considering the CABEB algorithm, the throughput on the Ethernet is slightly lower, reaching 9.446 Mbits/second This is divided fairly between the 2 stations, as shown in Table 6.1. There are no packets discarded by either of the stations. However, the collision rate with the CABEB algorithm is 100%, as seen by the two stations. With the CABEB algorithm, each station on completion of transmission collides with the loser of the previous contention and then backs-off 2 slots deterministically. The losing station who is on the 2nd collision backs-off 0 slots, again deterministically, since it was the winner for the previous packet. Thus, there is an alternation between the 2 stations when transmitting 1500 byte packets. Thus, every packet encounters a collision for the 1500 byte packet case, with the new CABEB algorithm. This contributes to the slightly lower throughput with the *new* algorithm.

When considering the mixture of the *new* and *old* stations, we see the throughput loss is reduced, and we achieve a throughput of 9.771 Mbits/second, compared to 9.806 Mbits/second with both the stations being *old*. The collision rate in fact comes down significantly for the *new* station, and it also achieves a higher throughput than the *old* station it is competing against. The packet discard rate does not drop substantially, although the *new* station experiences fewer discards than the *old* stations.

To understand the extent of *transient unfairness* of the standard BEB algorithm in comparison to our proposed CABEB algorithm, we show the performance of the UDP-like protocol with 2 and 3 stations in the network, in Table 6.2. One of the more interesting metrics is the number of consecutive packets transmitted by a station. With two stations, the CABEB algorithm has both the mean and maximum number of consecutive packets transmitted by a station (for a 5 second simulation run) of 1, showing that it has not only long term fairness but also transient fairness. In comparison, for the 64 byte packet case, the standard BEB algorithm has a mean of 2812 packets transmitted consecutively and (for the duration of the simulation) a maximum of 6626 consecutively transmitted packets. Although using the maximum is not generally used, we present it to show the potential for the extent of transient unfairness with the standard BEB algorithm.

Table 6.2 also shows the results with 3 identical nodes transmitting packets of 64 or 1500 bytes. The mean number of consecutive packets transmitted by a station reduces somewhat, as expected with 3 nodes, using the standard BEB - down to 1431.16 packets with 64 byte packets. The CABEB begins to allow multiple packets to be transmitted by a station, as there is a small increase in the probability that stations will go beyond 2 collisions and therefore approach the standard BEB behavior. A CABEB station transmits a mean of 7.42 packets consecutively in the 3 node, 64 byte packet case. The benefit of the CABEB reduces when we consider the 1500 byte packet case with 3 nodes. The 2 slot backoff is masked by the packet transmission time, and as a result, the CABEB algorithm with multiple stations is unable to consistently show an advantage over the BEB with large packets.

With 3 stations, the statistics for the access latency with the CABEB vs. the standard BEB are comparable, indicating that there is no penalty because of the CABEB algorithm relative to the access latency or *transient fairness*.

6.2 Performance of UDP-like datagram protocol with CABEB algorithm with multiple stations

In Figure 6.1, we show the overall Ethernet throughput with UDP when multiple stations are attempting to transmit as fast as possible on the channel. We vary the number of stations from 2 to 13 stations, transmitting fixed length packets of 64 or 1500 bytes. All the stations are either *old* stations using the standard BEB algorithm or the *new* CABEB algorithm. As the number of stations increases from 2 to 13, the overall achieved throughput goes down with both algorithms, as expected. With

1500 byte packets, the penalty paid for the fairness in access to the channel by the CABEB algorithm is relatively small, and reduces as we go to larger numbers of stations. When the number of stations goes to 13, the difference between the *old* and the *new* environments is small. However, when we observe the performance with the 64 byte packets, the penalty for the *new* station environment is quite substantial, and appears to not reduce too much with increasing number of stations.

Type of stations	**pkt. size (bytes)**	**Consecutive Pkt. Statistics.**	
		Mean # Pkts.	Max. # Pkts.
New-New	64	1.0	1.0
Old-Old	64	2812.8	6626.0
New-New	1500	1.0	1.0
Old-Old	1500	167.58	347.0
New-New-New	64	7.42	2882.0
Old-Old-Old	64	1431.16	3647.0
New-New-New	1500	78.0	202.0
Old-Old-Old	1500	77.1	159.0

Table 6.2: UDP Performance with New (CABEB) stations and Old (standard)stations. (5 second simulation run).

The primary reason for the reduction in throughput is due to an increase in the number of collisions. When we go to 13 stations, the percentage of collisions with the *new* CABEB algorithm is 23.5%, with 64 byte packets, while with the *old* standard BEB algorithm, the collision rate is 7.8%. With 1500 byte packets however, the difference in the collision rate is not as significant, since the *old* environment already suffers a collision rate of 81.4 %. The *new* CABEB algorithm suffers a collision rate of a little over 100% (every packet successfully transmitted as well as those discarded suffer a collision). Therefore, the degradation going to the CABEB backoff algorithm is not substantial for the 1500 byte packet case.

6.3 Performance of a TCP-like reliable, window flow-controlled protocol with CABEB

One of the primary reasons for introducing the CABEB algorithm is to enhance the performance of protocols such as TCP, where the capture effect substantially reduces the performance of such a window flow controlled, reliable transport protocol. The acknowledgments are delayed due to the capture effect while the transmitter has transmitted its complete window and thus the channel is idle for a long period of time. The CABEB algorithm overcomes this.

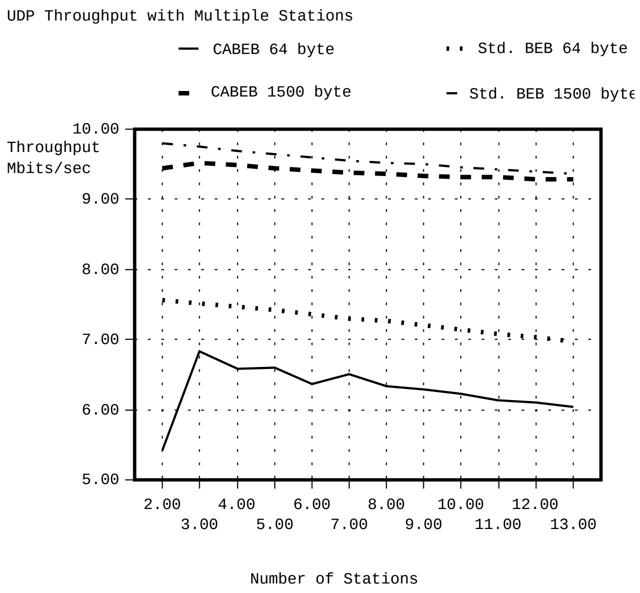

Figure 6.1: UDP Performance with multiple New (CABEB) stations and Old (standard)stations

Table 6.3 shows the performance with the CABEB algorithm or the standard BEB algorithm when the transmitter is sending packets of 64 or 1500 bytes, including all headers. The transmit window is 50 packets. The receiver generates an acknowledgment every other packet, and when an acknowledgment is received, the window is allowed to advance by 2 packets. The inter-arrival of packets to be transmitted at the source is sufficiently small so that the station is able to saturate the Ethernet. Acknowledgments are 64 bytes long, including all headers.

When both the stations are *new* stations using the CABEB algorithm, the overall throughput achieved is 4.59 Mbits/second, with the transmitting source getting a bandwidth of 2.772 Mbits/second. On the other hand, when both the stations are the *old* standard BEB algorithm, the overall throughput is only 3.915 Mbits/second, and the source gets a throughput of only 2.083 Mbit/second. The number of packets transmitted on the channel goes from 7646 packets/second with *old* stations to 8965 packets/second with *new* stations. The difference in the overall collision rate now is much smaller: 46.5% packets experience collisions in the *new* environment versus 30.5% collisions with the *old* environment. Thus, the CABEB achieves the desired effect of increasing the throughput of TCP.

Type of stn.	**pkt. size bytes**	**Enet tput (Mbps)**	**Station 1 (source)**		**Station 2 (sink)**	
			Thruput (Mbps)	Mean Access Time (μsecs)	Thruput (Mbps)	Mean Access Time (μsecs)
N-N	64	4.59	2.772	160	1.818	250
O-O	64	3.915	2.083	200	1.832	200
N-O	64	5.134	3.422	130	1.711	120
O-N	64	3.914	2.108	190	1.806	220
N-N	1500	8.604	8.404	260	0.2	750
O-O	1500	7.623	7.354	210	0.27	1600
N-O	1500	9.439	9.242	130	0.197	120
O-N	1500	7.6	7.324	200	0.275	200

Table 6.3: 2-node TCP Performance (Window=50) with New (CABEB) stations and/or Old (standard)stations (simulation time is 30 seconds.)

Another important characteristic to observe is the channel access latency. Because the window is only 50 packets wide, there is no extended capture of the channel possible by either of the stations (a station can at most send its full window of packets, and then has to wait for an ack.). But, when the sending station captures the channel for a period of time, the station acting as the sink has to wait for transmitting the acknowledgment and thus may see an increase in the mean access time. Even more importantly, because only the packet at the head of the queue of packets to be transmitted experiences the large access time (and the rest of the queued packets after that are able to get quick access, since the channel is either no longer captured or in fact may now be captured by this station), the variance in the channel access latency is important to observe. With the *new* stations, the mean access latency at the source is 160 μseconds, while with the *old* stations it is 200 μseconds. But even more striking is the variance in access latency at the source, which goes up by an order of magnitude, from 36 (milliseconds)^2 with the *new* stations to 570 (milliseconds)^2 with the *old* stations. This shows that even with a small window of 50 small packets, the amount of time the source has to wait when the sink (which sends 64 byte acks.) has captured the channel is substantial. This variability would make interactive applications, or other applications such as multimedia which require small variability in the response times, to behave unacceptably with the *old* stations.

When we go to a mixture of *new* (source of TCP) and *old* (sink of TCP) stations, the throughput achieved is even higher, going up to 5.134 Mbits/second. This occurs only when the *new* station is the source, so that its polite behavior of deferring by 2 slots on the first collision after a capture allows the sink to transmit the acknowledgment, and the overall data transfer makes better progress. In fact, the channel access latency characteristics show dramatic improvement. The mean access time reduces slightly, but the variance reduces by 1 to 2 orders of magnitude. This results in the substantial increase in throughput. However, when the *old* station is the source and the *new* station is the sink for TCP flow, the performance improvement is lost, since the capture of the channel continues to be achieved by the *old* station. The results are comparable to the case when both stations are the *old* standard BEB case.

Table 6.3 also shows the performance with 1500 byte packets for the different combinations of stations. We observe that the throughput increase with the *new* CABEB algorithm is more apparent. The throughput goes up from 7.623 Mbits/second with *old* stations to 8.604 Mbits/second with *new* stations, a 12.87% increase. What is also more revealing is the mean and variance in access latency. At the source and sink, the variance in access latency goes up by an order of magnitude. Furthermore, at the receiver, where the station suffers because of the capture of the channel, the mean access latency goes up by more than a factor of 2 from 750 milliseconds to 1600 milliseconds. As noted with the 64 byte packet case, the throughput in the mixed environment shows substantial improvement with the *new* (source) and *old* (sink) combination where the effect of capture is mitigated. There is no improvement when the source is an *old* station and the sink is a *new* station.

7 CONCLUSIONS

Ethernet has gained significant popularity and has seen widespread deployment and achieved a huge installed base in the industry. Despite many proposals for modifications and improvements, we believe Ethernet works well and has proven its interoperability and plug-and-play capability. Given the increased in network bandwidth demand due to faster computer systems, we see a trend towards proliferation of smaller work group with even dedicated Ethernets (for two nodes), to reduce the degree of sharing on the shared multiple access network. Also, new applications and protocols are beginning to place significant demand on the network, both in terms of throughput and latency. In this paper, we have studied a known performance issue with Ethernets, called the capture effect and its impact on transport protocol or applications that use window flow control.

The Ethernet capture effect is the behavior where one station transmits consecutive packets exclusively for a prolonged period of time, despite contention from other station(s). This is a known behavior of the standard Binary Exponential Backoff (BEB), when implementations follow the most aggressive behavior allowed by the standard. This behavior is most dramatic at high loads with a small number of active nodes. We presented the problem based on our measurement experiments with Alpha AXP systems running Digital's OSF/1 operating system and the TCP/IP protocol. We observed a throughput degradation caused by TCP's acknowledgments being held back due to the capture effect. Stations suffered long periods of backoff resulting in unnecessary idle periods on the channel. These are symptoms of the capture effect.

With a detailed description and analysis of the capture effect for the 2-node case, we showed that after a node has successfully transmitted a packet the node has a significantly higher probability of winning subsequent transmit opportunities.

We consider several performance metrics which are critical in understanding the efficacy of the backoff algorithms, including a notion of transient unfairness. The Ethernet capture effect contributes to dramatic transient unfairness. One direct measure of the capture effect is the number of consecutive transmit packets by a given station. Another measure of the capture effect is the packet discard rate due to excessive collisions.

We presented our solution to the Ethernet capture effect, called the Capture Avoidance BEB (CABEB) algorithm. Our primary considerations in developing the solution is to interoperate well with standard implementations and to be compliant with the Ethernet/802.3 standard. The CABEB is an enhanced BEB algorithm, where it uses BEB for all cases of collision resolution except for one. After a station has successfully transmitted a packet and if the channel is idle prior to the station's transmission of a second packet, the station uses an enhanced backoff algorithm for collision resolution for the second packet. In this case, if the second packet experiences a collision, it uses a fixed 2 slot backoff time for the first collision and a fixed 0 slot backoff for the second collision. For the third and subsequently collisions (for the second packet), it uses the standard BEB algorithm for backoff. For a two node network, we showed that this algorithm works in a deterministic manner where the two nodes alternate their transmissions..

We studied through simulations, the behavior of the BEB and the CABEB for 2 to 13 nodes with packet sizes of 64 bytes and 1500 bytes. We simulated the worst case environment for the network with infinite load from all nodes, maximum physical extent of the network, and minimum and maximum packet sizes. We use these to study the worst case behavior of the backoff algorithms, when the effect of capture is at its extreme. We observed that for a UDP-like datagram protocol the CABEB has absolutely no capture effect for a 2 node network. However, it pays a price for achieving this fairness in having a higher collision rate which results in lower throughput, especially for small packets. The BEB on the other other hand has significant period of capture indicated by the number of consecutive transmits. We showed that the mean number of consecutive transmits grows up to 2812 minimum size packets, and 167 maximum size packets. The discard rate due to excessive collision is also lower for the CABEB. We also presented the results of a network with a mixture of CABEB and BEB stations where the performance is much closer to that of a two node BEB.

We presented the performance of UDP-like datagram protocol for multiple stations. We observed that the throughput of CABEB for nodes 2-13 is lower than the BEB for 64 byte packets, when the percentage cost of a collision is the highest. However, the degradation in throughput for 1500 byte packets is not substantial. When we go to 6 nodes, the standard BEB still has transient unfairness, as we found a mean value for the number of consecutive transmits remain as high as 514.9 packets in contrast to 4.4 with the CABEB. Looking at the mean channel access latency, we observed that the CABEB is slightly higher than the BEB due to the collisions used by the CABEB for alternating the transmit opportunity. The 95th percentile for the channel access latency with the CABEB is consistently less than the BEB, which reflects the advantage gained by the reduction in capture of the channel.

We studied the performance of a TCP-like protocol and showed that the CABEB has a significant performance improvement over the BEB. For minimum size packets, we obtained 4.59 Mbits/s with the CABEB, in contrast to 3.915 Mbits/s with the BEB. For maximum size packets, we obtained 8.604 Mbits/s with the CABEB, in contrast to 7.623 Mbits/s with the BEB. For a mixture of stations (CABEB and BEB) where the CABEB is the data source, we get even higher throughput at 9.439 Mbits/s (maximum size packet) and 5.134 Mbits/s (minimum size packet). We believe this is due to the polite behavior of the source node (CABEB) in allowing the sink node (BEB) to return acknowledgments. When the source node uses the standard BEB however, we see little improvement. Thus, for a TCP-like protocol, there is a dramatic improvement in overall performance with CABEB, achieving our goal.

The CABEB is an enhanced BEB that is compliant and interoperates with the Ethernet/IEEE 802.3 standards. We believe that the simplicity in the enhancement to the BEB and the performance benefits shown make the CABEB an attractive improvement to CSMA/CD algorithm.

ACKNOWLEDGMENTS

We would like to thank the large number of people who helped us along the way by introducing us to the problem, help understand it and develop solutions for it. Tony Lauck and Bill Hawe need special thanks for a lot of guidance and help throughout this work. Chran-Ham Chang, Dave Cherkus, Mike Cochran, John Forecast, Heather Gray, Philippe Klein, Jeff Mogul, Ashok Nadkarni, Peter Smith, and Eric Werme all helped in the measurement analysis and short-term fixes in different ways. Our thanks to Tony Jordan, Dave Fite, Mike O'Connor, Avraham Menachem and Gady Daniely in enabling our ideas to be incorporated into reality. We would like to thank Paul Koning for reviewing the paper. The initial simulation of the Ethernet by Vasmi Abidi and Dye-Jyun Ma helped us get off the ground, and our thanks to them as well.

REFERENCES

1. ANSI/IEEE Standard 802.3-1985, "*Carrier Sense Multiple Access with Collision Detection*" IEEE, October 1985.
2. Boggs, D. R., Mogul, J. C., Kent, C. A., "*Measured Capacity of an Ethernet: Myths and Reality*", Proceedings of ACM Sigcomm '88 Symposium on Communications Architectures & Protocols, Stanford, CA., 1988 (Also as Computer Communications Review, Vol. 18, No. 4, August 1988).
3. Gonsalves, T. A., Tobagi, F. A., "*On the Performance Effects of Station Locations and Access Protocol Parameters in Ethernet Networks*", IEEE Transactions on Communications, Vol. 36, No. 4, April 1988: 441-449.
4. IEEE Fast Ethernet Standard, IEEE 802.3u Proposal for 100-Base-T Standard, IEEE Proposal, March 1994.
5. Leffler, S.J., McKusick, M. K., Karels, M. J., Quarterman, J. S., "*The Design and Implementation of the 4.3 BSD UNIX Operating System*", Addison-Wesley Publishing Company, May 1989.
6. Marathe, M., "*Design Analysis of a Local Area Network*", Proceedings of the Computer Networking Symposium, December 1980.
7. Marathe, M., Kumar, S., "*Analytical Models for an Ethernet-like Local Area Network Link*", Proceedings of the 1981 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, 1981.
8. Metcalfe, R. M., Boggs, D. R., "*Ethernet: Distributed Packet Switching for Local Computer Networks*", Communications of the ACM, Vol. 19, No. 7, July 1976, 395-404.
9. Postel, J. B., "*Internet Protocol*", RFC 791, SRI Network Information Center, Menlo Park, CA., September 1981.
10. Postel, J. B., "*Transmission Control Protocol*", RFC 793, SRI Network Information Center, Menlo Park, CA., September 1981.
11. Postel, J. B., "*User Datagram Protocol*", RFC 768, SRI Network Information Center, Menlo Park, CA., August 1980.
12. Schwetman, H.D., "*CSIM Reference Manual,*" MCC Technical Report, ACA-ST-257-87 Rev 14, March 1990.
13. Tobagi, F. A., Hunt, V. B., "*Performance Analysis of Carrier Sense Multiple Access with Collision Detection",* Computer Networks 4 (1980): 245-259.
14. Digital Internal Document: "*TGEC Specification*", Part Number:XX, 1991.
15. Digital Equipment Corporation, "*DECchip 21040 Ethernet LAN Controller for PCI: Data Sheet*", Digital Equipment Corporation Order Number: EC-N0280-72, Nov. 1993.
16. Digital Equipment Corporation, "*EtherWORKS Turbo PCI User Information*", Digital Equipment Corporation Order Number: EK-DE435-OM.A01, March 1994.
17. Advanced Micro Devices, "*Am7990 Local Area Network Controller for Ethernet (LANCE)*", AMD Ethernet/IEEE 802.3 Family World Network Data Book/Handbook, 1992.