Efficient and Accurate Ethernet Simulation

Jia Wang*and Srinivasan Keshav
Cornell Network Research Group (C/NRG)
Department of Computer Science, Cornell University
Ithaca, NY 14853-7501
{*jiawang, skeshav*}@*cs.cornell.edu*

Abstract

The Internet is increasingly being called upon to provide different levels of service to different applications and users. A practical problem in doing so is that although Ethernet is one of the hops for nearly all communication in the Internet, it does not provide any QoS guarantees. A natural question, therefore, is the effect of offered load on Ethernet throughput and delay. In this paper, we present several techniques for accurately and efficiently modeling the behavior of a heavily loaded Ethernet link. We first present a distributed approach to exact simulation of Ethernet, which greatly simplifies collision detection. Then, we describe an efficient distributed simulation model, called Fast Ethernet Simulation, that empirically models an Ethernet link to quickly and accurately simulate it. By eliminating the implementation of CSMA/CD protocol, our approach reduces computational complexity drastically while still maintaining desirable accuracy. Performance results show that our techniques not only add very little overhead (less than 5% in our tests) to the basic cost of simulating an Ethernet link, but also closely match real-world measurements. We also present efficient techniques for compressing cumulative distributions using hyperbolic curves and for monitoring the load on a heavily-loaded link.

1 Introduction

The Internet is increasingly being called upon to provide different levels of service to different applications and users. A practical problem in doing so is that although Ethernet is one of the hops for nearly all communication in the Internet, it does not provide any QoS guarantees. (New versions of Ethernet do provide guarantees, but there is a huge embedded base of 'legacy' installations that do not.) One might argue that these Ethernet hops are rarely the bottleneck, and thus their effect on communication is negligible. However, it is equally true that the Level-2 infrastructure in a typical site is rarely managed for performance. Thus, it is possible, and even likely, that a large fraction of Ethernet installations are overloaded from time to time. Our interest, therefore, is in determining the effect of this overload, and concommitent performance degradation, on application performance.

*This material is based upon work supported under a National Science Foundation Graduate Fellowship.

We study this interaction in two steps. In this paper, we present techniques for accurately and efficiently modeling the performance of a heavily loaded Ethernet link. In future work, we will report on the use of this technique to study the effect of Ethernet load on application performance.

Despite its widespread use, there is little knowledge about the behavior of Ethernet-like CSMA/CD LANs under heavy load. Analytical models tend to study performance based on over-simplified assumptions. However, this usually leads their results to be biased towards network performance under ideal conditions. Once the complexities of CSMA/CD, as described in the IEEE 802.3 standards, are introduced, such models become intractable [15] [17] [26] [28]. The inaccuracy and incompleteness of analytical work has led researchers in the past to resort to measurement and simulation to obtain meaningful results. Even these approaches are not without problems.

Actual measurements on a physical LAN require manual configuration of the network, which is expensive and cumbersome. Simulation is a better tool to obtain adequate information on functionality and performance of communication networks and protocols. However, to simulate the behavior of CSMA/CD, precise collision detection, packet loss (due to collision and buffer overflow), and packet transmission/retransmission need to be implemented in order to get accurate and valuable results. Sophisticated computation and complicated data structure manipulation make the traditional detailed simulation of CSMA/CD slow and complex, especially for crowded network and/or heavy loaded link.

Fortunately, the above two approaches are not the only choices we have to comply to get accurate Ethernet performance results. In this paper, we first present a distributed approach to exact simulation of Ethernet, which eliminates sophisticated collision detection. Then, we propose an efficient distributed simulation model, called Fast Ethernet Simulation, which models an Ethernet link empirically to quickly and accurately simulate it. By eliminating the implementation of CSMA/CD protocol, our approach reduces the complexity drastically while still maintaining desirable accuracy.

The remainder of this paper is organized as follows. Section 2 gives a brief summary of some related work on Ethernet link performance. An overview of our approach is stated in Section 3. We describe our simulation model of CSMA/CD in Section 4. Then, we discuss how the performance parameters are modeled based on the CSMA/CD simulation results and propose the Fast Ethernet Simulation model in Section 5 and Section 6, respectively. Some performance results are shown in Section 7 to demonstrate that the Fast Ethernet Simulation achieves the efficiency as well as the accuracy. Finally, we summarize our work in Section 9.

2 Related Work

Ethernet refers to a family of LAN multiple access protocols that vary in details such as bandwidth, collision detection mechanism etc. In this paper, we use Ethernet to mean an unslotted, 1-persistent, carrier-sense multiple access method with collision detection and binary exponential backoff. In the past two decades, Ethernet performance has been carefully studied ([1] [2] [3] [4] [5] [6] [7] [8] [9] [11] [14] [15] [17] [18] [19] [20] [21] [23] [24] [25] [26] [27] [28] [29]). Many analytical models have been formulated ([2] [3] [5] [6] [9] [14] [15] [17] [25] [26] [27] [28]). Due to the complexity of the CSMA/CD retransmission algorithm and variety of LAN topologies, these analytical approaches employ a number of simplifying assumptions, such as balanced-star configuration, finite populations, unimodal or constant packet lengths, small packet size, and no buffering to obtain tractable results of Ethernet performance. However, it is not clear how relevant these are to the actual performance of Ethernet. For instance, analytical results show that the maximum achievable throughput with CSMA/CD is 60% [28]. In fact, the CSMA/CD protocol, as implemented in practice, can achieve throughput of 97% for large packet size [4]. Smith and Hain [24] also presented results of experiment measuring Ethernet performance using station monitoring, which show that measured performance differs significantly from predictions made by typical analytical models. Since none of existing analytical models is applicable and sufficient to estimate the real Ethernet performance, it is difficult to conduct accurate performance evaluation by strictly analytical means. Simulation and/or measurement are necessary to obtain accurate and adequate information on the Ethernet performance.

In recent years, several Ethernet performance studies have been based on detailed simulation and/or measurement to avoid some of the simplifying assumptions mentioned above. Gonsalves [7] presented performance measurements on operational 3 and 10 Mbit/s Ethernet to explore how packet size and offered load affect the link throughput and packet delay. Boggs, Mogul and Kent [4] also presented measurement of behavior of an Ethernet under varying combinations of packet lengths, network lengths, and number hosts to show that Ethernet is capable of good performance for high-bandwidth applications, especially when response time is not closely constrained. However, due to the inflexibility of measurement on physical networks, only limited performance measurements under typical network configuration are reported in literature. Other measurement work can be found in [8] [23] [24].

By using simulation, performance can be easily measured for various Ethernet topologies and system configurations. Some detailed simulation models are presented in [11] [18] [19] [20] [21] [29], which can be used to model Ethernet with different size, transmission rate, Ethernet length and station distribution, etc. An event driven simulation model is the standard approach. In such a model, the movement of packets in the model is expressed in terms of events. A global table is maintained to record each event that takes place at a specified time. We distinguish between two types of event-driven simulation models. In a *centralized* approach, the medium is simulated by an active entity that keeps track of packets sent by each station, and informs each station about the current state of the medium. The centralized medium also detects and computes the exact time a collision occurs and sends out jam signals. Each station need only model packet transmission and/or retransmission due to collision and packet drop due to buffer overflow. Although these detailed simulation models may achieve accurate performance results, they are too complex. In this paper, we present an efficient alternative approach, that we call *distributed* simulation. We will present more detail about this model in Section 4.

To sum up, existing analytical models tend to be over-simplified, existing measurement work is too cumbersome to replicate, and existing simulation techniques are computationally inefficient. In this paper, we present an efficient approach to exact Ethernet simulation, and a new technique for even faster simulation, which we call Fast Ethernet Simulation. We validate our work by comparing our performance prediction to real-world measurements. We also show that our techniques are computationally much more efficient than those proposed in the past.

3 Our Approach

Figure 1 summarizes our approach, which consists of the following steps.

1. designing and validating a detailed CSMA/CD simulator;
2. collecting and modeling performance measurements using this simulator;
3. creating a fast simulator;
4. validating the fast simulator.

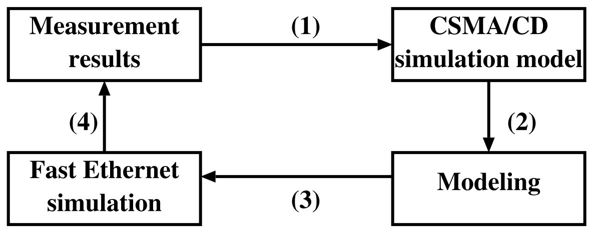

Figure 1: Overview of our approach.

Our first step was to build a detailed simulation of CSMA/CD using the REAL network simulator [22]. This allowed us to reproduce a variety of workloads, workstation configurations, and card buffer sizes without the practical difficulties of dealing with a real testbed. Additionally, by comparing simulator results with experimental measurements, we made sure that the output of the simulations was valid. This first step, therefore, bought us flexibility, even though it was computationally expensive. Unlike other simulations of CSMA/CD, we did not perform sophisticated computation for collision detection. This made our simulator both simpler and more accurate. A detailed description of our approach is in Section 4.

With this simulator in hand, we were able to generate a large number of empirical performance measurements corresponding to a variety of configurations. The second step was to reduced this information into a compact model. The model, described in Section 5, reflects the dependencies of link throughput and packet delay on offered load, packet size and buffer size of host adapter cards.

The third step was to exploit the compact performance model to develop Fast Ethernet Simulation. The key idea here was to predict performance using a simple computation on the compact model of past empirical measurements. We came up with two new techniques in traffic monitoring and performance prediction in the course of this work. First, we developed an efficient technique for statistical estimation of the load on a link over a specific time interval. Second, we used a family of hyperbolic curves to represent the cumulative distributions of delay. Details of these techniques are presented in Section 7.

Finally, in the fourth and the last step, we validated the results obtained from Fast Ethernet Simulation with that obtained using the detailed simulation. The results presented in Section 7 show that, by eliminating the implementation of CSMA/CD protocol, our fast simulation model reduces the complexity drastically while the simulation results still achieving desirable accuracy.

4 Distributed CSMA/CD Simulation

Our first step is to create an accurate simulation of CSMA/CD. This allows us to generate performance data for network configurations and workloads that are hard to create in an actual testbed. We validated the accuracy of the simulator by comparing the performance metrics obtained from our simulator with those reported in the literature.

Before describing our simulation, we first give a brief review of the 1-persistent Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol. Assume that n stations attach to the Ethernet link (Figure 2). A station senses the medium before sending a packet and sends the packet immediately after the medium is idle. If a collision is detected while sending a packet, the station sends out a jam signal and exponential backoff scheme is employed. Upon a collision, the station waits for a random time chosen from the interval [0, 2 $\times$ max propagation delay] before retransmitting the collided packet. If retransmission fails, the station backs off again for a random time chosen from the interval with double length of the previous one. Each subsequent collision doubles the backoff interval length until the retransmission succeeds (the backoff interval is reset to its initial value upon a successful retransmission of packet). If the backoff interval becomes too large (e.g. after 16 retransmission), the packet is dropped and the backoff interval is reset.

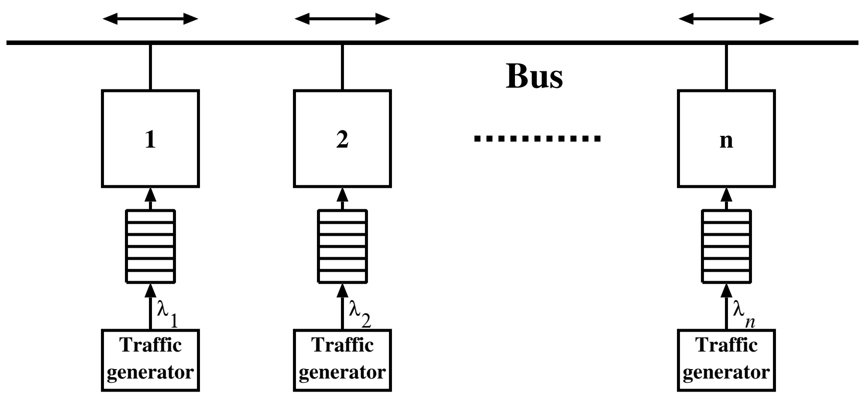

Figure 2: Model configuration for the distributed simulation

Most existing simulations of CSMA/CD model the transmission medium as a centralized active entity. This entity determines the exact time at which each station knows that a packet has been placed on the medium, or that a collision has occurred. Determining these times is non-trivial because multiple packets can be placed on different parts of an Ethernet nearly simultaneously. Indeed, it turns out that to accurately determine these times, the simulation has to correctly model the electromagnetic propagation of data signals on the medium. This makes it algorithmically complex and hard to correctly implement. After several attempts at creating an accurate CSMA/CD simulation model using this approach, we realized that an alternative approach elegantly solves the modeling problem. In this approach, the medium is passive, not active. Instead, each station on the Ethernet acts as a router, forwarding packets from an incoming link to an outgoing link. An idle station that receives a packet changes its state to busy. If a packet arrives at a busy station, a collision is detected and the station broadcasts a jam indication to the other stations. In our approach, therefore, the stations cooperate to jointly simulate the medium. This makes the simulation both easy to program and easy to validate. The next subsection describes this approach in greater detail.

4.1 State Diagram

We model CSMA/CD using the station state diagram shown in Figure 3. It can be seen that the medium is not modeled as an active entity. Instead, stations exchange data, jam, and collision messages as would happen in an actual Ethernet. Each simulated station is responsible for actions such as packet transmission/retransmission, collision detection, signaling.

A simulated station can be in one of the seven states: idle, sending, receiving, wait-for-backoff-end-and-jam-end, wait-for-jam-end, wait-for-backoff-end, and receiving-and-wait-for-backoff-end.

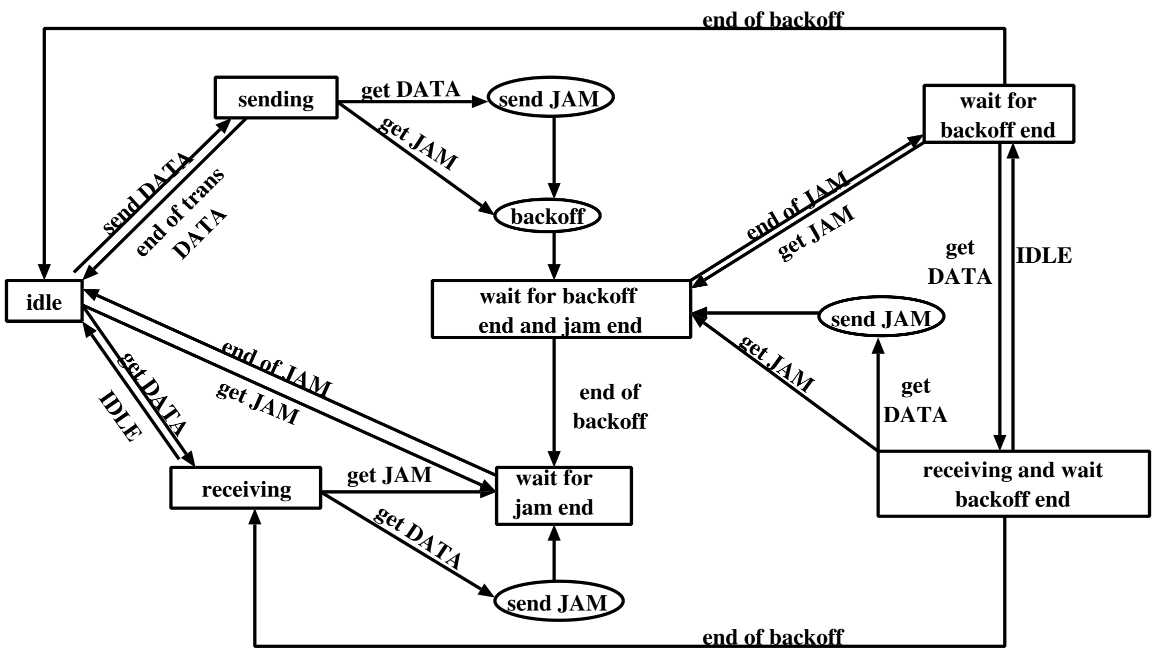

Figure 3: State diagram of distributed CSMA/CD simulation

Assume the simulated LAN has a topology as shown in Figure 2. Packet propagation on the Ethernet is emulated as consecutive propagation by the intermediate stations towards the destination, i.e. when a node gets a packet from one of its neighbors, it sends the packet to its other adjacent neighbor. Propagation delay on a link is modeled by setting the transmission delay on the point to point link between adjacent stations to be the delay on the medium between two adjacent stations. An IDLE signal is sent on the link to identify the end of a DATA packet or JAM signal. Collisions can then be detected by a station if it receives a DATA packet or JAM signal while sending or receiving DATA packets. On detecting a collision, a JAM signal is sent to all the other stations. These stations automatically receive the JAM signal after the appropriate propagation delay. As should be clear, with this approach, no computation is needed to determine the set of stations that involved in the collision and the exact time they know of the collision.

4.2 Experimental results

We implement this simulation model on REAL Simulator [22]. The Ethernet link we simulated is 10BaseT. In order to validate our CSAM/CD simulator, we use the same Ethernet configuration and system workload (Table

1) as that used in Gonsalves's measurements [7]:

Bus bandwidth	10 Mbit/s
Max propagation delay	30 μs
Jam time after collision	32 bits (= 3.2 μs)
Slot size	512 bits (= 51.2 μs)
Buffer size for each station	1 packet
Idle period is uniformly distributed	
Packet size P is fixed for all stations	

Table 1: Ethernet configuration and system workload in Gonsalves's measurement.

Gonsalves used a closed-loop system in his performance measurement work, i.e. after completion of transmission of a packet, a station waits for a random period, with mean θ, before the next packet is queued for transmission in its buffer. He claims that the offered load of station i, G_i, is defined to be the throughput of station i if the network had infinite capacity, i.e., $G_i = T_p/\theta_i$, where $T_p = P/C$, P is packet length and C is the capacity of the Ethernet link. The total offered load G of N stations is given by $\sum_{i=1} NG_i$. However, when measuring the performance of Ethernet, we believe that the offered traffic load should be *independent* of the packet transmission, i.e. the entire measurement system should be an open-loop system. In order to compare our simulation results to the measurement results, we adopt Gonsalves's closed-loop system model in the validation of our simulator, the collect performance measurements from an open-loop model.

The performance results of measurement and simulation are shown in Figure 4, respectively. We consider two performance parameters: packet delay and link throughput.

Delay: The packet delay is defined to be the time it takes to successfully send a packet, measured from the time the host puts the packet into the sending queue. We look at both mean packet delay and cumulative delay distribution for typical packet length and offered load. Figure 4(a) and (c) show the measurement and simulation results of mean packet delay as a function of total offered load, in 10Mbit/s, for various value of P. The results obtained from our simulation match that presented in Gonsalves's measurement reasonably well. For example, when $P = 1500$ bytes and $G = 300\%$, the mean packet delay reported by Gonsalves's measurement is 19.6 ms, while our simulation results show that it would be 20.9 ms. The cumulative delay distributions are also given in Figure 4(d) for $P = 512$ bytes under offered load 30%, 90%, and 300%. Our simulations do not match Gonsalves's results too well for small packet sizes and light load (see Figure 4(c)). However, in this range the absolute values of delays are small, and though the relative error is significant, the absolute value is not. Moreover, our simulations consistently over-estimate the delay, so that our performance predictions are conservative.

Throughput: We define the link throughput to be the link goodput, i.e. the number of bytes that are successfully transferred during a time unit. Figure 4(b) shows the variations of total throughput with total offered load for $P = 64$, 512, 1500 bytes, which are obtained from measurement and simulation. Under high offered load, the link throughputs are measured as 26%, 70%, 82% for $P = 64$, 512, 1500 bytes, respectively. In our simulation, the corresponding throughputs are 36%, 71%, and 83%, which are very close to the measurement results of the actual system.

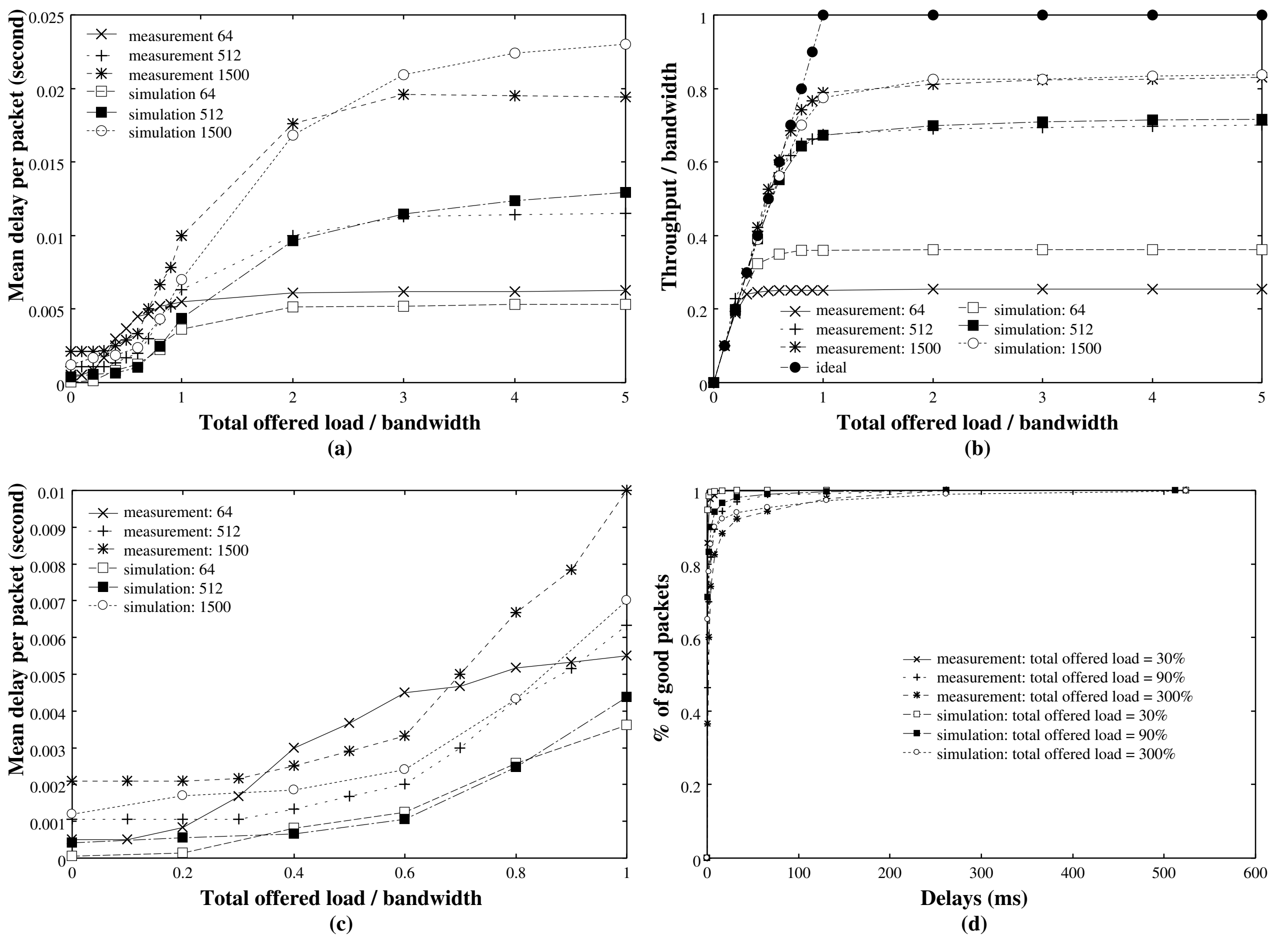

Figure 4: 10 Mbps Ethernet measurement and simulation results (under the same configuration in Gonsalves' work) on mean packet delay and link throughput. (a) delay versus total offered load, (b) throughput versus total offered load, (c) zoomed version of (a) for total offered load $\leq$ 100%, (d) cumulative delay distributions of total offered load = 30%, 90%, and 300%: packet size = 512 bytes.

The differences between Gonsalves's measurements and ours are due to a differences in the node configurations. The measurement results were obtained for a single specific configuration (i.e. spacing between stations) that is not described in the paper. Our results, instead, are the average over an ensemble of configurations. The performance results generated by our simulator match his results in terms of both packet delay and link throughput for heavy loads and large packet sizes. Since this is the region of interest in our work, we claim that our simulator realistically models Ethernet.

After validation, we used our distributed CSMA/CD simulator to generate a large number of performance measurements corresponding to a variety of configurations. Typical simulation parameters are set according to the IEEE 802.3 specification [12]. However, the system configuration that we simulated is different from the one we used in the simulator validation in three ways.

First, we used an open-loop system to model the Ethernet environment, i.e. the process of traffic generation is independent of the process of packet transmission. Figure 5 compares the corresponding simulation results obtained for the corrected configuration and Gonsalves's configuration. Although the performance results do not differ significantly from the results of closed-loop system, we believe it better models reality. Therefore, we adopt the open-loop system configuration in generating simulation results for the compact model described in Section 5.

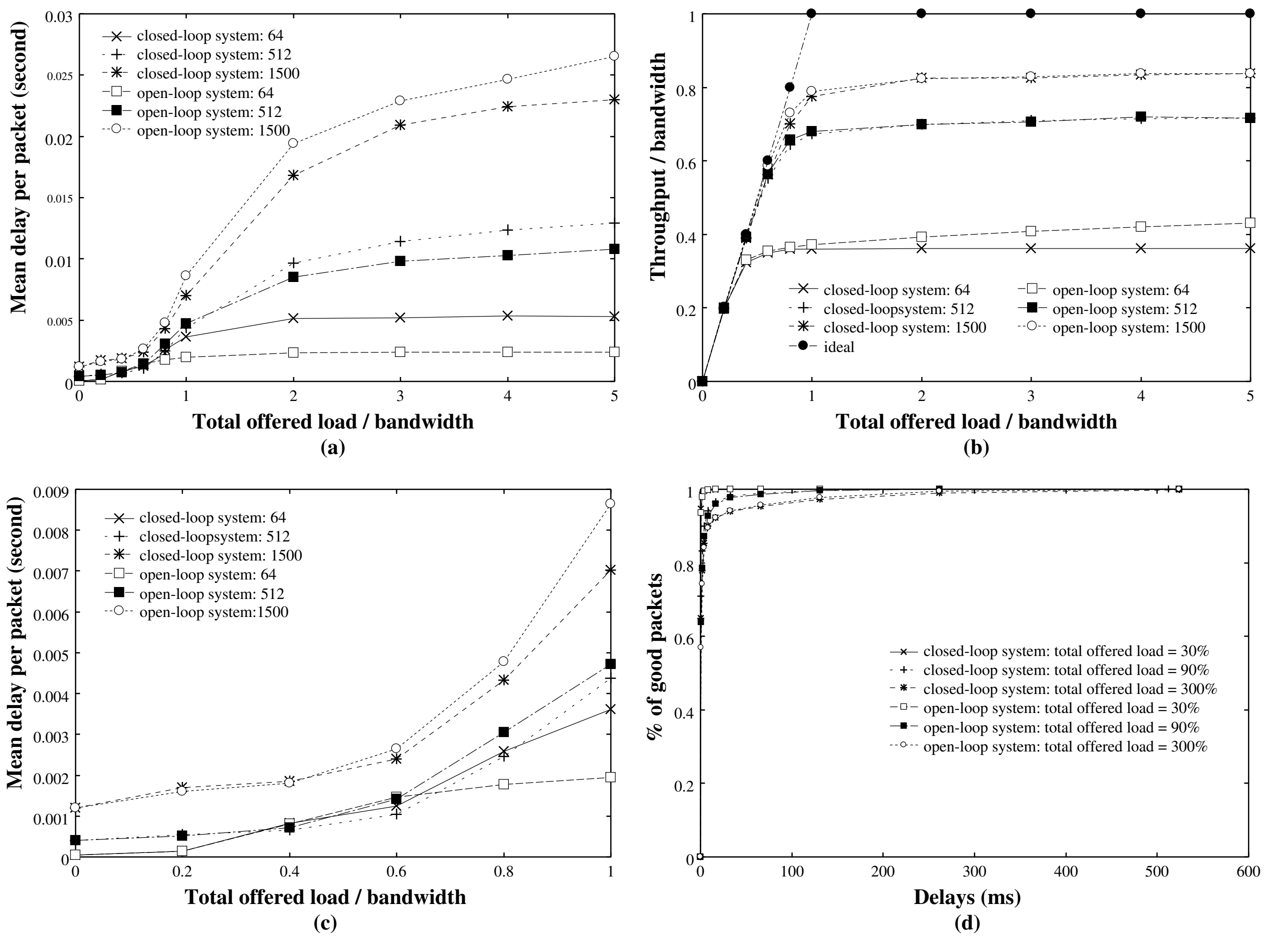

Figure 5: 10 Mbps Ethernet simulation results under open-loop and closed-loop configurations. (a) delay versus total offered load, (b) throughput versus total offered load, (c) zoomed version of (a): total offered load $\leq$ 100%, (d) cumulative delay distributions of total offered load = 30%, 90%, and 300%: packet size = 512 bytes.

Second, Gonsalves's measurements were done based on an assumption that the buffer size of each station is one packet. This is not true in the real world. Each station may have a fixed number of buffers to hold packets waiting for transmission. Packets that arrive for transmission when this buffer is full are discarded. Multiple buffers have a non-negligible impact on the system performance. As the buffer size increases, fewer packets are dropped due to congestion. The mean queueing delay of packets are also increased significantly. Figure 6 shows variations of the mean packet delay and link throughput for buffer size = 1, 4, 8 packets.

The mean packet delay increases approximately proportional to the buffer size when the link offered load is high. We can simply consider each station as a single-queue-single-server queueing system where the single queue is the buffer and the single server is the Ethernet. Let the average system service time s be the average time to successfully transfer one packet, measured from the time the host first acquires the channel. Then s can be approximated as the average packet delay when buffer size = 1 packet. Therefore, the average number of packet

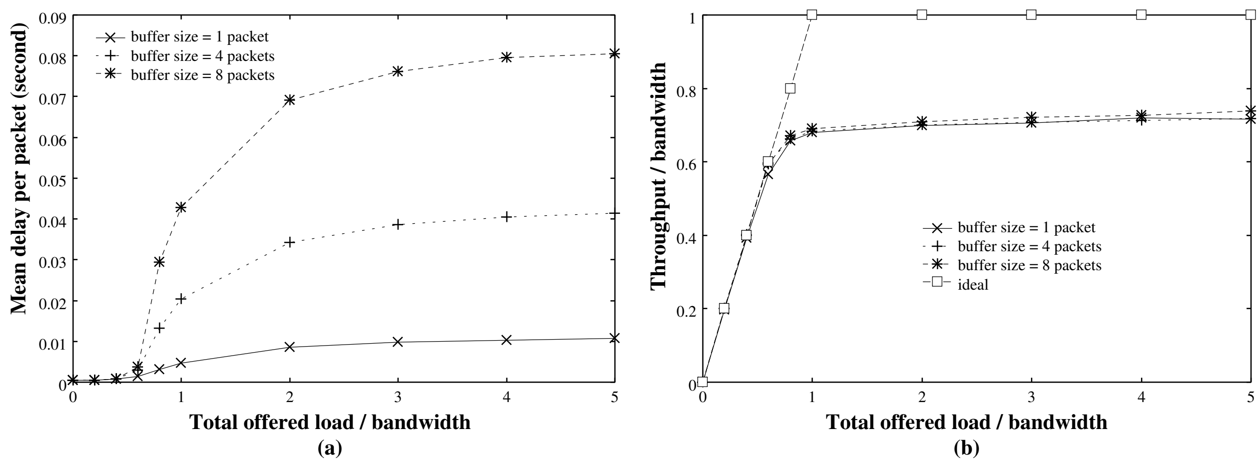

Figure 6: 10 Mbps Ethernet simulation results with buffer size of 1, 4, 8 packets (packet size = 512 bytes): (a) mean delay versus total offered load, (b) throughput versus total offered load.

in the queue is $N = \lambda \cdot s$, where λ is the arrival rate of each station. Thus,

$$packet\ delay = \begin{cases} N \cdot s & if\ N\ <\ buffer\ size \\ buffer\ size\ \cdot\ s & if\ N\ \geq\ buffer\ size \end{cases}$$

If N is greater than the buffer size, then the mean packet delay increases approximately proportional to the increment of buffer size. However, the simulation results show that the total throughput is not affected much by the buffer size even when the offered load is high.

Third, Gonsalves chose packet interarrival times at each station from a uniform distribution. It is not clear that the uniform distribution correctly models Ethernet workload. Indeed, Willinger et al[30] have shown that the heavy-tailed Pareto distribution is probably a better model of reality. It occurred to us that before choosing any particular distribution, it was necessary to determine the degree to which the packet interarrival time distribution determines Ethernet performance in the first place. To do so, we studied Ethernet performance while choosing packet interarrival times from the uniform, Poisson, normal, and Pareto distributions. We also looked at two types synchronized workloads. In the synchronized in-phase workload, packets are generated at each station at the exactly same time. This leads to the worst case for the Ethernet performance. Correspondingly, in the synchronized out-of-phase model, packets are generated at each station with the maximum possible spacing given a particular offered load. This is the best case for Ethernet. These two cases represent the upper and lower bounds of the performance that Ethernet can achieve under different workloads for the same value of the offered load. Finally, we introduce the notion of a skewed synchronized workload. With a skew factor k ($0 < \mathrm{k} < 1$), packets are randomly generated at each station within the first k fraction of each randomly selected time period. The smaller the skew factor k, the more contention the stations incur on the Ethernet link, and the closer the workload is to the worst case.

Figure 7 shows the mean packet delay and link throughput as the function of link offered load for different workloads. First, notice that, except for the synchronized workloads, there are only slight differences in performance with different workloads. This means that one might as well choose a uniform or Poisson packet interarrival model, because the performance with either workload is more or less the same. The reason for this is that under light load, all the workloads see few collisions and small absolute delays. So, the packet interarrival time distribution does not change the delay distribution very much. Under heavy load, each station almost always has a packet to send in its send buffer. This decouples performance from the details of the packet interarrival process. Second, although with synchronized in-phase traffic the performance is poor and significantly different from that achieved by Poisson traffic, this workload is rare in reality. Moreover, performance with this workload becomes similar to that achieved with Poisson traffic even with a small skew factor of $k = 0.1$. This means that unless the workload exhibits perfect in-phase synchrony, an event with very small likelihood, the achieved performance is close to that with Poisson traffic. Our experimental results therefore indicate that Poisson packet interarrivals adequately model the workload for Ethernet traffic. Given this result and the fact that generating a specific offered load with Poisson arrivals is much easier than to do so with Pareto arrivals. From now on, we will assume that the packet interarrival process is Poisson.

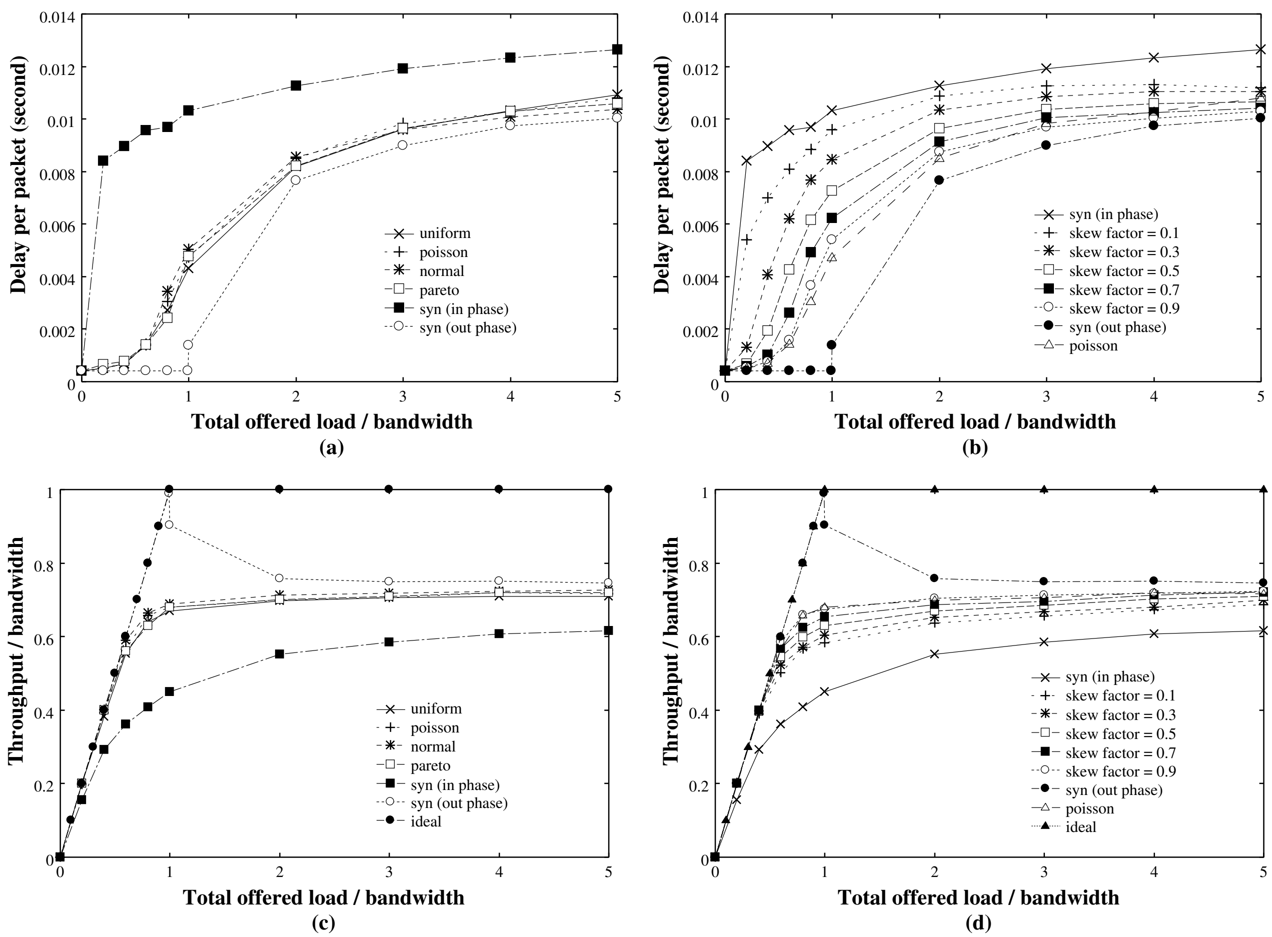

Figure 7: 10 Mbps Ethernet simulation results (under different workload) on mean packet delay and link throughput: packet size = 512 bytes, buffer size = 1 packet.

5 Modeling

The second step in our approach is compactly model Ethernet performance for all possible network configurations. We achieve fast simulation by referring to this model to predict performance for a given configuration. By examining the results of the detailed simulation, we found that the two important performance parameters, packet delay and link throughput, are functions of three independent variables: the mean packet size, the total link offered load, and the buffer size for each station. We explain this next.

5.1 Throughput

We found that link throughput is a monotonically increasing and piecewise linear function of the link load and mean packet size (Figures 4 and 5). We measure and store the link throughput for a sequence of packet sizes and link offered loads. In order to increase the lookup table granularity in regions where the throughput changes rapidly as a function of the independent parameters, we choose the distance between consecutive points in the sequence to be inversely-proportional to the slope of the throughput curve at that point. Then, for a given mean packet size and link offered load, the link throughput is obtained by a linear 2-dimensional interpolation between the adjacent stored values in the lookup table.

5.2 Delay

Instead of storing the mean delay for a given configuration, we chose to model the *cumulative distribution* of delays achieved for a given setting of independent parameters. This is because even with the same workload, different packets may experience different delays due to the randomization inherent in the Ethernet protocol and other stations' behavior. A pure prediction of mean packet delay is not enough to capture this variation. During fast simulation, for a specific packet, we generate a delay as a random variable drawn from this distribution. Modeling the cumulative instead of the density allows us to trivially generate a random variable from this distribution. However, naively storing the cumulative delay distribution requires too much storage. We need a way compress this information, choosing the compression scheme such that rapid decompression is possible. A family of well-known hyperbolic curves turns out to satisfy this requirement.

5.2.1 Using hyperbolic curves to model cumulative delay distributions

Consider the family of hyperbolic curves represented by

$$y = \frac{x + kx}{1 + kx},$$

where $k = tan(\frac{\pi}{2}\alpha) - 1$, and $\alpha \in [0, 1]$. An interesting characteristic of this family is that the single variable α controls shape of the curves. Figure 8(a) shows some curves for different values of control variable α.

The cumulative delay distribution curves, as shown in Figure 5(d) are very similar to these hyperbolic curves. (Similar cumulative distributions of delay are also reported in Gonsalves's measurement results [7]). Thus, we decided to use hyperbolic curves to model the cumulative delay distributions. The advantage of this approach is that a cumulative distribution curve can be "compressed" into single variable α. This makes the lookup table extremely compact. Moreover, optimal values of α can be chosen as the least-squares fit to the actual distribution. Because of the single control variable α, the least-squares fitting is much more easily determined than with multiple control variables. An example of such modeling is shown in Figure 8(b).

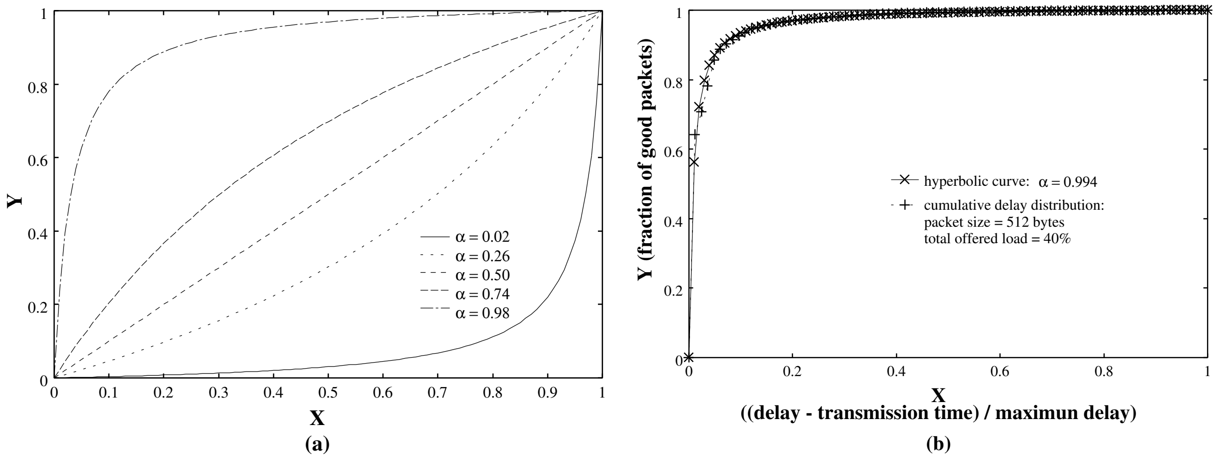

Figure 8: Hyperbolic curves: (a) A group of hyperbolic curves, (b) using hyperbolic curve to model cumulative delay distribution.

To sum up, instead of storing a cumulative distribution curve, we store the value of α in the lookup table as a function of the mean packet size and offered load. We use the same indexing and interpolation techniques to compute the cumulative delay distribution for typical packet size and offered load as we did to compute link throughput. The value of α corresponding to each combination of mean packet size and link offered load listed in the indexing sequences is stored in the lookup table. Before interpolation, four cumulative delay distributions are first computed according to the α values of adjacent indexed packet sizes and offered loads in the lookup table. Then, linear 2-dimension interpolations are applied on these cumulative distributions to compute the cumulative delay distribution for the given mean packet size and link offered load.

6 Fast Ethernet Simulation

Fast simulation is achieved by predicting performance metrics based on the compact model described earlier. Since there are no collisions and backoffs, this implementation of CSMA/CD protocol is much faster.

6.1 Approach

The simulation configuration is shown in Figure 9. Several stations attach to a shared Ethernet link. Each simulated station has three active components: traffic generator, traffic monitor and performance predictor. Recall that the input to the performance prediction model is the offered load and the mean packet size. The performance monitor measures these parameters on the fly and feeds them to the performance predictor. The performance predictor determines whether or not the packet has chance to go through the link and if so, how much the delay it going to suffer. Finally, packet delivery is simulated according to the predicted performance information. We describe traffic monitor and performance predictor in more detail next.

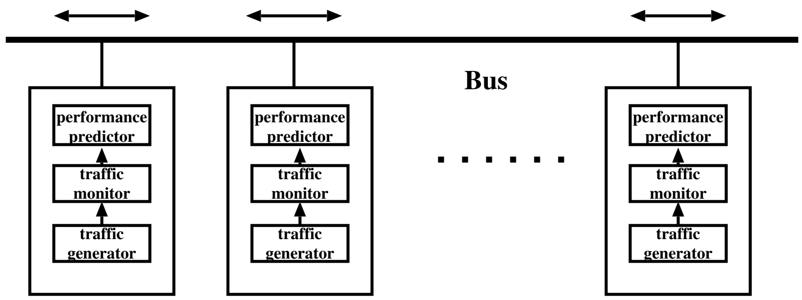

Figure 9: Fast Ethernet simulation configuration.

6.2 Monitoring Statistical Information

In order to predict Ethernet performance, we need to monitor the mean packet length and mean total offered load over some time period. We use a dynamic window scheme to compute the mean values of packet length and total offered load. The size of time window should represent a balance between sensitivity to the current system state and stability of the measurement. If the size of time window is too large, then it will mix up two different patterns of traffic load on the link. On the other hand, if the time window size is too small, then the control mechanism will react to a transient burst of packet arrivals, which makes the system unstable. We choose the time window size to be 1 second, because it is long enough to even out busty traffic, but not so long as to lose slow-scale changes in traffic.

A naive way to monitor the traffic through a link would be to keep a list of active packets that are transmitted within the current time window. Upon each packet arrival, we add the new packet to the head of the list and remove from the end of the list the inactive packets (i.e., packets transmitted before the beginning of current time window). The statistical information is also updated accordingly. However, this algorithm can be expensive because the traffic monitor needs to update its statistical information upon each packet arrival. Moreover, updating the active packet list can be time consuming. In the worst case, traversing the entire list is necessary to remove the inactive packets. The work done by the algorithm increases as the offered load increases. Thus, the computing complexity increases at least linearly with the traffic load. If simulated link is heavy loaded, traffic monitoring will incur a big computational overhead. Therefore, a more efficient monitoring algorithm is desirable.

We have designed a 'ring buffer' approach to speed up the monitoring process. The structure of the ring buffer is shown in Figure 10. Time is divided into equal-size slots. Each slot records the traffic statistics information during that time slot. Let W be the size of time window, T_s be the size of the time slot and w be the number of slots within one time window, then $W = w \cdot T_s$. At end of each time slot, the window is shifted one slot forward and the overall traffic information is updated by removing the information from the newly invalidated slot and adding the information from the newly validated slot. We choose $T_s = 1024\ \mu s$ to make the computation of the current slot index efficient. The current slot is determined purely by using integer binary operations such as masking and shifting. For instance, if the current time is $m\ \mu s$, then the slot index is computed as (m & 0x000ffc00) $\gg 10$.

With the ring buffer monitoring technique, no active packet list is maintained, which makes the information updating very efficient in terms of both time and space, requiring only a constant time overhead during heavy load. Under lightly loaded conditions, the window may move several slots, increasing the overhead. However, this overhead is incurred when the packet simulation overhead is small, so we do not consider it to be a great burden.

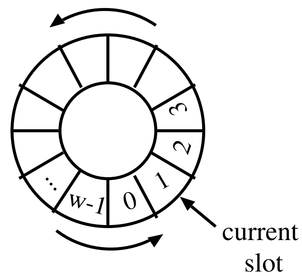

Figure 10: The ring structure to monitor the traffic statistics of Ethernet link.

6.3 Predicting Performance

The traffic load information monitored by the traffic monitor is passed to the performance predictor. The major inputs for the performance prediction are mean packet size, mean total offered load, and buffer size of each station. The performance predictor determines the packet delay. The packet delay is randomly chosen according to the cumulative delay distribution corresponding to the current mean packet size and total offered load. As we mentioned in Section 4, performance results show that the packet delay increases proportional to the increase in the buffer size of each station. The exact impact of multiple buffers on the performance is very hard to model. We need balance the trade off between the efficiency and accuracy. We model the delay simply by multiplying the buffer size with the corresponding delay predicted for the one buffer case. Although it may incur some inaccuracy to our performance modeling, especially for low offered load cases (i.e. it may over-estimate the delay a packet suffers when the offered load is relatively low), the absolute error is still relatively small and acceptable. Thus, the performance predictor can be considered as a function of mean packet delay, total offered load and station buffer size.

$$\text{delay} = f(\text{mean packet size, total offered load, buffer size})$$

Finally, the packet delivery is simulated according to the predicted performance information. The packet is discarded if delay $= \infty$.

7 Performance Evaluation

In this section, we compare the performance results obtained from Fast Ethernet Simulation with that obtained from the detailed CSMA/CD simulation. We implemented the fast simulation on the REAL simulator [22] similar to the way we implemented the detailed CSMA/CD simulation.

As mentioned earlier, we adopt an open simulation model for traffic generation, i.e., the generation of packets is independent of the delivery of packets. For the simulation results shown here, we assume packets are generated from a Poisson distribution. However, our results are independent of the Poisson assumption. Packet destinations are randomly chosen from a uniform distribution.

We first examine the accuracy of the fast simulation model. Figure 11 compares performance results obtained from fast simulation with the detailed CSMA/CD simulation for P = 64, 512, 1500 bytes and buffer size = 4500 bytes. The link throughputs match with each other perfectly, whereas the mean packet delays have small error due to our simple modeling of Ethernet performance for multiple buffers. This is particularly evident for the case where the mean packet size is small (e.g. P = 64 bytes), where the same absolute value of the buffer size corresponds to a relatively larger number of queued packets. When P = 512 bytes and the total offered load is 200%, the mean packet delays observed on the detailed CSMA/CD simulation and fast Ethernet simulation are 69 ms and 65 ms, while the corresponding throughput are 70.8 % and 70.9 %, respectively. Thus, this figure indicates that fast simulation accurately models the performance of an actual Ethernet network.

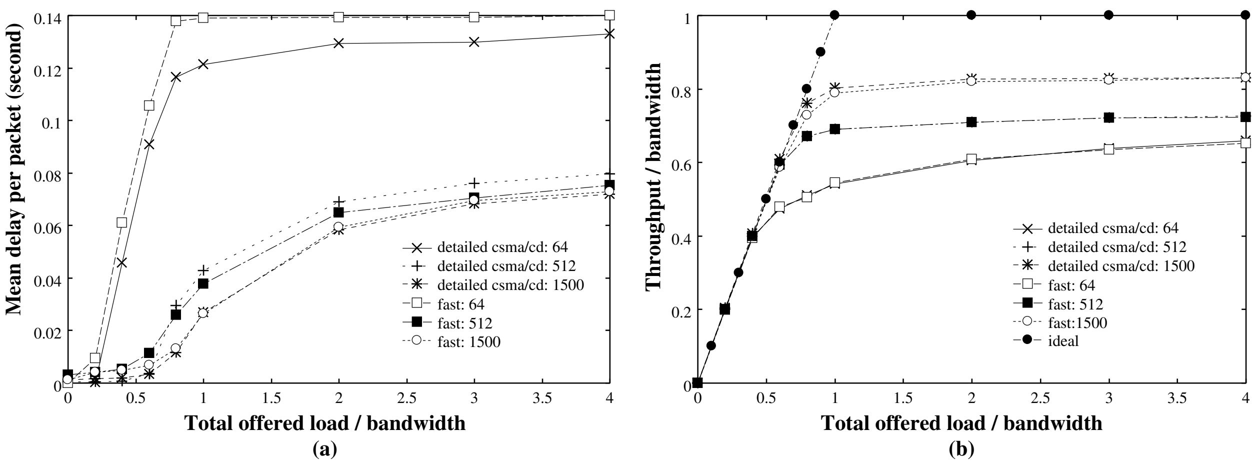

Figure 11: The simulation results for fast Ethernet simulation and CSMA/CD simulation for packet size = 64, 512, and 1500 bytes and buffer size = 4500 bytes: (a) delay versus total offered load, (b) throughput versus total offered load.

The fast simulation approach imposes two overheads: traffic monitoring and performance prediction. In order to measure this overhead, we introduce an ‘easy-fast’ Ethernet simulation model. In this model, instead of passing monitored traffic information to the performance predictor, the user can manually set the link traffic load and mean packet length. The performance prediction is done purely based on these pre-setup traffic information and the predicted Ethernet performance is independent of the traffic going through the simulated link. In this case, the simulated stations do not need to send the amount of packets specified by the offered load to the Ethernet link to make the traffic monitor “see” the traffic load and tell the performance predictor this information. Easy-fast simulation does not require traffic monitoring, so by comparing the simulation time for easy-fast simulation with the simulation time for fast Ethernet simulation, we can determine the overhead for traffic monitoring. Similarly, by comparing the time taken for easy-fast simulation with a base case simulation with no performance prediction, we can estimate the overhead for performance prediction. (Note that our easy-fast simulation model provides a back-door for defining the traffic load independent of the actual number of packets placed by a station on a link. This is extremely useful in situations where we want to study the effect of a highly loaded link on a particular application *without* actually generating the loading cross traffic.)

Thus, we examine the overhead incurred by each component of fast simulation model by comparing the time complexities of the following four simulation models of LAN: (a) base simulation model: simulation of LAN (e.g. traffic generation, packet delivery) without employing any Ethernet protocol, (b)easy-fast simulation model: simulation of LAN with user-defined traffic information and performance prediction, (c) fast simulation model: simulation of LAN using traffic monitoring and performance prediction, and (d) detailed CSMA/CD simulation model.

We ran these four LAN simulation models under the same system configuration to simulate 10 seconds behavior of Ethernet on Solaris. The CPU time measured for these four simulation models as a function of total offered load is shown in Figure 12. Figure 13 shows the corresponding slow down ratios as function of total offered load. For instance, when total link offered load is 100%, the four different simulation models take 18.01 seconds, 18.74 seconds, 18.90 seconds, and 296.27 seconds, respectively, the corresponding slow down ratios are 1.0, 1.04, 1.05, and 16.45. The CPU time consumed by the traffic monitor and the performance predictor take 1% and 4% of that consumed by the base model, respectively, which are very small portions of the total simulation time. The major portion of simulation time is due to factors other than the simulation of Ethernet delays, e.g. traffic generation and packet delivery, as we observed from base model. Thus, we believe that our fast simulation model doesn't add noticeable overhead to the simulator. The total CPU time consumed of detailed CSMA/CD simulation would be 1645% of that of the base model. This shows that the exactly simulation of CSMA/CD protocal (i.e. collision detection, packet retransmission, and signaling, etc.) is very time-consuming, even with distributed simulation. Although we do not present results for centralized simulation for CSMA/CD, we believe these to be another order of magnitude greater. Combining the results presented in Figure 11, 12, and 13, we claim that, by eliminating the exact implementation of CSMA/CD protocol, our fast simulation model reduces the complexity drastically while the simulation results still achieving desirable accuracy.

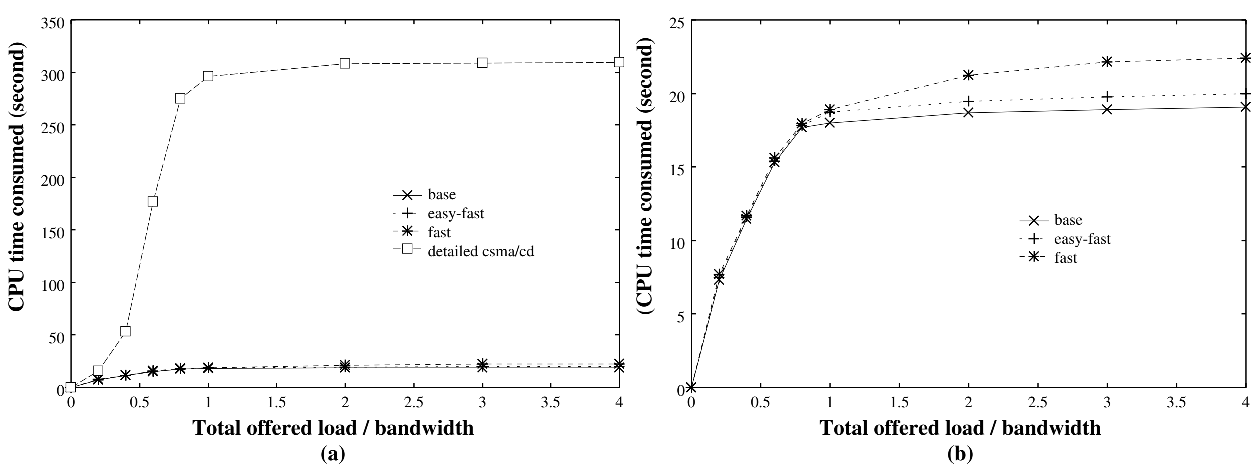

Figure 12: The CPU time consumed for 10 seconds simulations of 10 Mbps Ethernet link with packet size = 512 bytes and buffer size = 8 packets ((b) is the zoomed version of the three curves in the lower portion of (a)).

8 Conclusion

In this paper, we present two innovations. First, we describe a distributed approach to exact simulation of Ethernet, which eliminates the overhead of sophisticated collision detection. Second, we propose an efficient

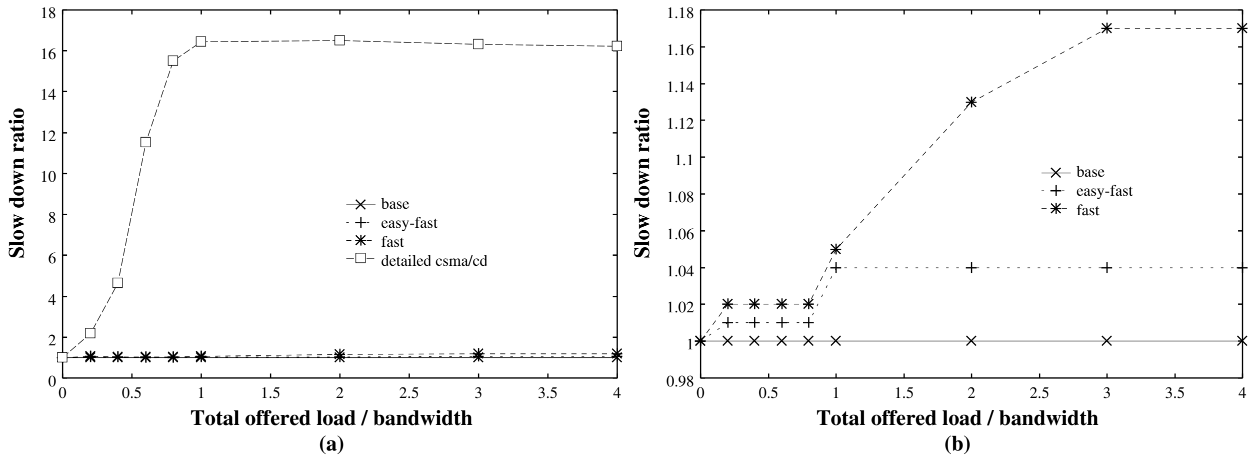

Figure 13: The ratio of CPU time consumed for 10 seconds simulations of 10 Mbps Ethernet link with packet size = 512 bytes and buffer size = 8 packets ((b) is the zoomed version of the three curves in the lower portion of (a)).

distributed simulation model, called Fast Ethernet Simulation, which empirically models an Ethernet link to quickly and accurately simulate it. Our work shows that by eliminating the exact implementation of precise collision detection, signaling, and packet retransmission, the time complexity of Ethernet simulation can be significantly improved while still maintaining simulation accuracy. Our detailed performance results demonstrate the accuracy and efficiency of our approach.

We came up with three new techniques as part of this work. First, we use a family of hyperbolic curves to represent the cumulative distributions of delay as a function of the offered load and mean packet size. Second, we present a near-constant time algorithm for monitoring the load on a link over a specific time interval. Third, we studied the impact of different link workloads on Ethernet performance and show that, although the Pareto distribution is the most realistic model of Ethernet traffic, Poisson is good enough to achieve accurate performance results besides being much easier than Pareto to be used in performance measurements. We believe that these techniques can be used in a variety of other situations, and represent new additions to the network protocol designer's toolbox.

The top level goal of our work is to study the effect of Ethernet load on application performance. Using the easy-fast simulation approach, we can subject traffic from an application to an desired Ethernet load with practically no additional performance overhead. This will allow us to study this interaction with unprecedented ease. We plan to present the results of this study in future work.

References

[1] R. Ayani, Y. Ismailov, M. Liljenstam, A. Popescu, H. Rajaei, and R. Ronngren, Modeling and simulation of a high speed LAN, Simulation, pp. 7-14, January 1995.

[2] G. T. Almes and E. D. Lazowska, The behavior of Ethernet-like computer communications networks, Proceedings of the 7th Symposium on Operating Systems Principles, pp. 66-81, December 1979.

[3] W. Bux, Local-area subnetworks: a performance comparison, IEEE Trans. on Communications, Vol. COM-29, No. 10, pp. 1465-1473, October 1981.

[4] D. R. Boggs, J. C. Mogul, and C. A. Kent, Measured capacity of an Ethernet: myths and reality, Proceedings of the SIGCOMM'88 Symposium on Communications Architectures and Protocols.

[5] E. Coyle and B. Liu, Finite population CSMA/CD networks, IEEE Trans. on Communications, Vol. COM-31, No. 10, pp. 1247-1251, November 1983.

[6] E. Coyle and B. Liu, A matrix representation of CSMA/CD networks, IEEE Trans. on Communications, Vol. COM33, No. 1, pp. 53-64, January 1985.

[7] T. A. Gonsalves, Measured performance of the Ethernet, Advances in Local Area Networks, pp. 383-410.

[8] R. Gusella, A measurement study of diskless workstation traffic on an Ethernet, IEEE Trans. on Communications, Vol. 38, No. 9, pp. 1557-1568, September 1990.

[9] T. A. Gonsalves and F. A. Tobagi, On the performance effects of station locations and access protocol parameters in Ethernet networks, IEEE Trans. on Communications, Vol. 36, No. 4, pp. 441-449, April 1988.

[10] F. Halsall, Data Communications, Computer Networks and Open Systems (4th edition), Addison-Wesley, 1995.

[11] H. D. Hughes and L. Li, Simulation model of an Ethernet, Computer Performance, Vol. 3, No. 4, pp. 210-217, December 1982.

[12] IEEE Standard 802.3 - Carrier Sense Multiple Access with Collision Detection (CSMA/CD), 1985.

[13] S. Keshav, An Engineering Approach to Computer Networking, Addison-Wesley, 1997.

[14] L. Kleinrock, and F. Tobagi, Packet switching in radio channels: part I - carrier sense multiple-access modes and their throughput-delay characteristics, IEEE Trans. on Communications, Vol. COM-23, No. 12, pp. 1400-1416, December 1975.

[15] S. S. Lam, A carrier sense multiple access protocol for local networks, Comput. Networks, Vol. 4, pp. 21-32, Feb. 1980.

[16] W. E. Leland, M. S. Taqqu, W. Willinger, D. Wilson, On the self-similar nature of Ethernet traffic (extended version), IEEE/ACM Trans. on Networking, Vol. 2, No. 1, pp. 1-15, February 1994.

[17] R. M. Metcalfe and D. R. Boggs, Ethernet: distributed packet switching for local computer networks, Commun. Ass. Comput. Mach., Vol 19, pp. 395-404, July 1976.

[18] P. Marino and A. del Rio, An accurate and fast CSMA/CD simulator, Microprocessing and Microprogramming, Vol. 39, No. 2-5, pp. 187-190, Dec. 1993.

[19] P. J. P. O'Reilly, and J. L. Hammond Jr., An efficient simulation technique for performance studies of CSMA/CD local networks, IEEE Journal on Selected Areas in Communications, Vol. SAC-2, No. 1, pp. 238-249, Jan. 1984.

[20] K. Prasad and R. Patel, Performance analysis of Ethernet based on an event driven simulation algorithm, Proceedings Conference on Local Computer Networks, 1988.

[21] K. Prasad and A. Singhal, Simulation of Ethernet performance based on single server and single queue model, Proceedings of the 12th Conference on Local Computer Networks, Oct. 1987.

[22] RealEdit v1.0, URL: http://www.cs.cornell.edu/skeshav/real/overview.html.

[23] J. F. Shoch and J. A. Hupp, Measured performance of an Ethernet local network, Communications of ACM, Vol. 23, No. 12, pp. 711-721, December 1980.

[24] W. R. Smith and R. Y. Kain, Ethernet performance under actual and simulated loads, Proceedings of 16th Conference on Local Computer Networks, pp. 569-581, 1991.

[25] S. Tasaka, Dynamic behavior of a CSMA-CD system with a finite population of buffered users, IEEE Trans. on Communications, Vol. COM-34, No. 6, pp. 576-586, June 1986.

[26] F. A. Tobagi and V. B. Hunt, Performance analysis of Carrier Sense Multiple Access With Collision Detection, Comput. Networks, Vol. 4, pp. 245-259, Oct./Nov. 1980.

[27] X. Tan, S. C. Jue, E. Lo, and R. H. S. Hardy, Transient performance of a CSMA system under temporary overload conditions, Proceedings of 15th Conference of Local Computer Networks, 1990.

[28] H. Takagi, and L. Kleinrock, Throughput analysis for persistent CSMA System, IEEE Trans. on Communications, Vol. COM-33, No. 7, pp. 627-638, July 1985.

[29] L. Y. Tsui and O. M. Ulgen, On modeling local area networks, 1988 Winter Simulation Conference Proceedings, pp. 842-849.

[30] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level, Proceedings of Sigcomm'95, pp. 100-113.