On the Behavior of Ethernet: Are Existing Analytic Models Adequate?

by

Speros Armyros

Technical Report CSRI-259
February 1992

Computer Systems Research Institute
University of Toronto
Toronto, Ontario, Canada
M5S 1A4

The computer Systems Research Institute (CSRI) is an interdisciplinary group formed to conduct research and development relevant to computer systems and their application. It is an Institute within the Faculty of Applied Science and Engineering, and the Faculty of Arts and Science, at the University of Toronto, and is supported in part by the Natural Sciences and Engineering Research Council of Canada.

Abstract

In this report, we have developed a detailed Ethernet simulation model whose accuracy has been validated against published measurement data from real Ethernet configurations. Our simulator has the additional capability of embedding the Ethernet subsystem in some simple closed queueing networks that can be modeled easily using Mean Value Analysis (MVA) techniques. Our motivation here is to assess the sensitivity of high-level MVA models of Ethernet-based distributed systems to the approximations made in existing analytical models, in particular the Flow Equivalent Service Center of Almes and Lazowska. After studying the operating characteristics of the protocol in a number of open and closed systems, we have concluded that the existing CSMA/CD analytic models do not capture the behavior of the Ethernet protocol except under light loads. The discrepancies are due mainly to the operation of the binary exponential backoff algorithm, which becomes a predominant factor under high loads. However, our efforts at finding a better analytic model were unsuccessful. Several other experiments are carried out, notably the measurement of inter-packet time distribution, and evaluation of a simple protocol modification designed to reduce collisions in heavy traffic.

We extend the above simulation model to emulate the operation of the novel Virtual Time CSMA with Collision Detection (VT-CSMA/CD) protocol, considering the same distributed computing environments as for the Ethernet. We construct an accurate MVA-based analytic model for the protocol in the context of closed queuing networks. VT-CSMA/CD appears to be more predictable because it does not rely so much on the retransmission strategy. Furthermore, we found that VT-CSMA/CD supports all systems under study with substantially better performance than the Ethernet.

Acknowledgments

I would first like to express my sincere appreciation to my advisor, Prof. Mart Molle for shortening the arduous and lengthy job of finding a thesis topic and for his instructive, enthusiastic and insightful guidance.

I would also like to extend my gratitude to Prof. Ken Sevcik for introducing me to the area of Computer Systems Modeling, supporting my work by his scholarly criticism and sound advice.

I feel obliged to Prof. Sognian Zhou for providing me access to the facilities of the Distributed Systems Lab at LP372. Special thanks goes to Harjinder Sigh Sandhu and Spiros Mancoridis for taking the time to help me deal with incorporating all these numerous figures into this document.

On a more personal note, I would like to thank all of my many friends who made my graduate life in Toronto so enjoyable and epecially to my girlfriend Alison Hsieh.

Contents

List of Figures

Chapter 1

Forward

Local Area Networks (LANs) have received a lot of attention in recent years, due to the shift in the structure of computer systems architecture. The older prototype, of a centralized architecture built around a mainframe as the centerpiece of computation, has lost a lot of popularity. During the last decade we have seen the advent of engineering workstations, personal computers, intelligent peripheral devices, and much more, all with the capability of performing their respective tasks independently of any supervisory machine. This decentralization of computing power has brought about the need to interconnect the different nodes, so as to fulfill the requirements for sharing of expensive resources (such as printers, plotters and file servers) and information (via common databases), and for high level services such as electronic mail. Thus, the current trend is the design of highly distributed environments, where sets of powerful workstations are accessing various types of servers (situated within short geographical distances) in order to support their computing activities. Connections of this sort require high-bandwidth cabling that LANs can provide.

The challenge for LANs has been to provide facilities that meet users' communication needs at a reasonable cost, with reliability, fairness and an acceptable level of performance. This requires efficient control of user accesses to the physical medium that interconnects the components of the LAN. A broad class of medium access protocols is the class of *contention* protocols. In those protocols which are based on random access techniques, each station accesses the channel, using partial information about the activity of the other stations. Because the various stations are physically displaced within the area that the network encompasses, destructive interferences between messages transmitted by two or more stations may occur. These interferences are called *collisions* and are usually resolved using a *retransmission feedback algorithm* (*backoff* algorithm), which attempts to "serialize" the message transmissions. The stability of this algorithm is crucial for network performance, especially for protocols that heavily depend upon the backoff algorithm.

A popular form of contention protocols is the access control scheme referred to as *Carrier Sense Multiple Access* (CSMA), which has been shown to be highly efficient for environments with short propagation delays. This scheme is essentially a generalization of the ALOHA broadcast network concepts [1]. In pure ALOHA all stations independently transmit their packets to the "master" station without regard to the other stations' signals. The random transmissions, the lack of central control and the long propagation delays, result in high collision rates which yield a maximum channel efficiency of only 18.4%.

On the other hand, CSMA techniques assume a communication medium such as coaxial cable, twisted pair or radio channel, where the carrier sensing feature is available. *Carrier Sense* means that any station wishing to transmit "listens" first. If the channel is idle, the station transmits its packet immediately. If the channel is busy (i.e., some other station is transmitting) the station will reschedule the packet transmission according to the type (*persistent* or *non-persistent*) of the particular CSMA protocol. *Multiple Access* implies that any station wishing to transmit can do so. No central controller is needed to decide who is able to transmit and in what order. This is also referred to as distributed control environment where all stations on the network are peers with equal access.

CSMA protocols can be *non-persistent* or *persistent* [2]. Persistent techniques include the cases of *1-persistent* and *p-persistent*. The distinction is made based on the action of a station when the line is sensed busy or idle. In the case of non-persistent CSMA, upon sensing a busy channel the station acts as if a collision had taken place. "Retransmission" is scheduled after a random time interval, when the channel will be sensed again. The disadvantage of this policy is that, after the present transmission there is a good chance that the channel will turn idle. An obvious alternative is to sense the channel constantly and transmit when the channel is sensed idle again. This is the so-called "1-persistent" which is a special case of the "p-persistent" strategy. According to the latter, stations having messages, transmit with probability p when the channel becomes idle. If a station does not transmit a message, it waits for an interval equal to the round-trip propagation time and senses the channel again. If it is busy the station repeats the foregoing algorithm. This scheme may avoid the conflict that would be guaranteed in the 1-persistent case, if there are at least two stations deferring transmission during the busy period.

In certain cases the feature of *Collision Detection* is available, which leads to higher channel utilization. Each station monitors the channel for a collision during packet transmission. This can be done by checking whether the station's output signal matches the one on the channel. In the event of a collision, colliding stations continue transmitting for a fixed period of time to ensure that all stations in the network detect the collision. This is also known as *jamming*. Transmission of the colliding packets is aborted.

The operation of the CSMA protocols can be either *Synchronous (slotted)* or *Asynchronous (unslotted)*. In the synchronous version all stations are synchronized on a master clock and packets can only begin transmission on a specified clock interval. In the asynchronous version each station executes a local copy of the distributed protocol and packet transmissions may start at any time independent of the operation of the other stations. Slotted protocols are harder to implement but easier to analyze, whereas unslotted ones have more practical value, although their analyses are usually intricate.

Xerox's implementation of the CSMA technique is one of the most popular, successful and marketable architectures: the *Ethernet* [3], which has also become an IEEE standard. Numerous studies have been done to investigate the Ethernet performance characteristics under a wide variety of circumstances. Many articles have been published so far, referring to specific systems running different applications or describing sophisticated methods for system tracing or artificial generation of workloads. Their goals are to evaluate the Ethernet's contribution to performance by means of measurements, simulation or analytical modeling. Apart from these sophisticated techniques, we should be able to apply a simpler methodology, in order to give answers to these performance related issues with modest amount of effort, provided that ample information about the system parameters is available.

In terms of this research, we have carried out several sets of simulation experiments in order to study the behavior of the Ethernet. Our goal is to document the operating characteristics of Ethernet in sufficient detail and, in the second place, to study its integration into large scale distributed environments (such as client/server configurations) to investigate the limiting behavior of the Ethernet when it becomes a performance bottleneck. Some of the experiments were used to validate our model against published measurement data for real system configurations that use Ethernet. Others were intended to identify unusual behaviors that cannot be explained by the existing models, particularly *product-form queuing networks* that include an Ethernet as a service center.

Since Ethernet falls into the category of CSMA protocols, its performance characteristics heavily depend upon the behavior of its retransmission algorithm. Thus, it is interesting to compare it with other CSMA protocols where randomness is achieved by more sophisticated mechanisms. A typical representative of the latter category is the class of Virtual Time CSMA (VT-CSMA) protocols [4, 5, 6]. In these protocols the decision on the transmission time of a packet is based on its arrival time and the history of changes of the channel status. The transmission scheduling is implemented by means of a second clock at each station which runs only when the channel is idle. This protocol achieves avoidance of the initial collision of packets in most cases, thus reducing the effect that the backoff algorithm has on the delay experienced by a packet.

The comparison of the Ethernet protocol and the VT-CSMA protocols, particularly its Collision Detection mode (VT-CSMA/CD), is done by extending the Ethernet simulation model so as to mimic the behavior of the latter protocol. A series of tests in the same system context, enable us to obtain the performance measures for the two protocols and identify the impact of the backoff algorithm in both cases.

In Chapter 2, we provide an extensive survey of the plethora of analytic studies relevant to Ethernet-like networks. A review of measurement studies gives a flavor of the network traffic in real Ethernet configurations. Since the techniques that we apply in order to model the systems considered in this thesis, are based on Queuing Network Modeling, we introduce the fundamental tools and the terminology of this modeling discipline. Finally, we present several models that have been developed for the cases of distributed computing environments, built around Ethernet interconnection.

In Chapter 3, we give the specifications of the system that we have modeled, after a detailed presentation of existing Ethernet implementations. A top-down approach of the simulation model, along with lower-lever implementation details is illustrated.

In Chapter 4, we present the results of the Ethernet simulation model. After having validated this model, we use it to investigate the Ethernet behavior in (i) open systems with Poisson arrivals, (ii) closed systems representable analytically by closed product-form queuing networks and (iii) client/server systems. We conduct a study of the inter-departure time distribution in the case of the open systems, in order to investigate the suitability of the Poisson input traffic model in these systems. In the case of the closed systems we study a simple modification of the Ethernet protocol that enhances its performance under heavy network traffic. We also describe some unsuccessful efforts towards a better analytic model of the Ethernet subsystem.

In Chapter 5, we extend the previous simulation model in order to represent the VT-CSMA/CD protocol. We validate the model against accurate analytical studies, and use it to determine the performance of the protocol in closed systems. For this case we develop an accurate analytic model that proves the predictability of the protocol. In all cases we compare the behavior of the Ethernet and VT-CSMA/CD. Overall, our results show the superiority of the latter scheme.

Finally, in Chapter 6, the thesis concludes with a concise review of the results.

Chapter 2

Overview of Previous Studies

This chapter provides a brief survey of publications related to the experiments carried out for this thesis. Several important issues about Ethernet-like local area networks are illustrated, including previous analytical modeling techniques as well as the measured performance of real Ethernet configurations. A similar but less elaborate presentation of the Virtual Time CSMA protocol is given. Finally, a brief introduction to the principles and terminology of Queuing Network Modeling, followed by the description of models for Ethernet-based distributed computing environments, completes this discussion.

2.1 Ethernet and Virtual Time CSMA protocols

In this section we will present two instances of the CSMA mechanisms that we defined in the introduction, namely *Ethernet* and *Virtual Time CSMA*. The former is widely used in a variety of distributed computing environments and applications, whereas the latter, despite its better throughput-delay performance than existing CSMA algorithms, has not made it to the commercial market yet. We will provide the operational characteristics of the above protocols and survey theoretical studies relevant to their performance that have been carried out.

2.1.1 Ethernet: Description and Analysis

Ethernet [3] is a system for local communication among computing stations, based on packet broadcasting technology, which ensures reliability and flexibility. The interconnection of the stations is achieved by means of a tapped coaxial cable and a media access control scheme, which allows multiple users to share a single broadcast bus in an orderly and fair manner. The shared communication medium (Ether) is a passive high speed channel. The access protocol is *asynchronous, 1-persistent CSMA with collision detection*, often abbreviated to CSMA/CD in the literature. The network topology is typically a *linear bus* or an *unrooted tree,* but in certain cases it can be regarded as a *balanced star.*

Since the stations are physically displaced from one another, two or more of them may sense that the channel is idle and start transmitting simultaneously, causing a collision. When a transmitting station detects a collision, it (i) aborts transmission, (ii) jams the channel for a short period j (by transmitting a short burst of noise) notifying all other stations to that effect, and then (iii) schedules its packet for retransmission after a random interval is chosen according to a *retransmission feedback* or *backoff* algorithm. The randomness is essential to avoid repeated collisions among the same set of stations.

The Ethernet uses a *truncated binary exponential* backoff algorithm. In the binary exponential backoff algorithm, the mean retransmission interval is doubled with each successive collision of the same packet. However, there can be an arbitrarily large delay before a packet is transmitted, even if the network is not heavily loaded. To avoid this problem, in the Ethernet the doubling of the interval stops after the first 10 successive collisions, based on the fact that the Ethernet specification permits a maximum of $2^{10} = 1024$ stations in the network [7]. After 16 consecutive collisions, the transmission attempt is abandoned and the packet is said to be *dropped.*

Formal analyses of Ethernet-like networks are numerous and provide a wide variety of results for various combinations of the appropriate parameters. They are based on the theory of stochastic processes and introduce certain assumptions in order to keep the analysis tractable. In many cases, however, these assumptions may not reflect real situations. The purpose of the above analytical works is to predict system performance usually in terms of *throughput* (the average number of packets that are successfully transmitted per second) and *average delay* (the average time it takes to deliver a packet starting from the instant the packet became available for transmission). The maximum achievable throughput is often called *capacity.*[1] We will try to give a representative range of such studies attempting to extract the important lessons from them.

In the original Ethernet paper Metcalfe and Boggs derive a simple set of formulas for characterizing the performance of an "experimental" 2.94 Mbits/sec Ethernet [3]. This analysis is extended by Almes and Lazowska in [8], where a more "complete" analytic model for Ethernet-like communications networks is presented.

According to that model, the network time is divided into three categories: *idle* periods during which no station desires to use the Ether, *transmission* periods during which the Ether has been acquired for successful packet transmission, and *contention* intervals during which a station attempts to obtain control of the Ether in order to transmit a packet. The model makes the simplifying assumption that time is divided into *slots* of length equal to the round-trip propagation delay. The above slot length is imposed by the fact that a

[1] In the literature, *throughput* is sometimes defined as the fraction of bandwidth used for transmission of useful information. If we adopt this definition, then the capacity is in the range $[0, 1]$.

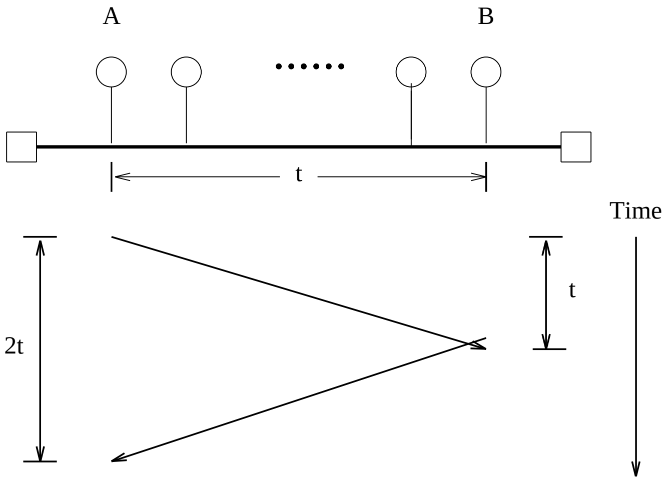


Figure 2.1: Worst case of detection of collision

collision is detectable within the round-trip propagation delay of the network, in the worst case. This is illustrated in Figure 2.1 where, in a bus network with end-to-end propagation delay of τ, station B begins to transmit dt time before the signal from station A arrives.

We define *instantaneous load*, Q, on an Ethernet to be the number of stations that desire to transmit a packet during a slot. The Ethernet protocol attempts to maximize the probability that exactly one station transmits during a slot. In particular, its backoff mechanism tries to dynamically estimate the number of contending stations and randomize them in such a way so as to map only one attempt per slot and therefore reduce the number of wasted slots. Metcalfe and Boggs [3] compute the probability that exactly one station attempts a transmission in a slot and therefore acquires the Ether. If Q stations compete for granting access to the Ethernet, and each station is transmitting with probability $1/Q$ independently, then the above probability equals:

$$A = (1 - \frac{1}{Q})^{Q-1} \tag{2.1}$$

Then an estimate of the number of contention slots prior to acquisition of the Ether by some station is:

$$W(Q) = \sum_{i=0}^{\infty} i \cdot A \cdot (1-A)^i = \frac{1-A}{A} = (\frac{Q}{Q-1})^{Q-1} - 1 \tag{2.2}$$

Given average packet size P bits, network carrying capacity C bits/sec, and slot time $S = 2\tau$ seconds, the *efficiency* (throughput) of the Ethernet for a given Q, $E(Q)$, is:

$$E(Q) = \frac{P/C}{P/C + W(Q) \cdot S} \tag{2.3}$$

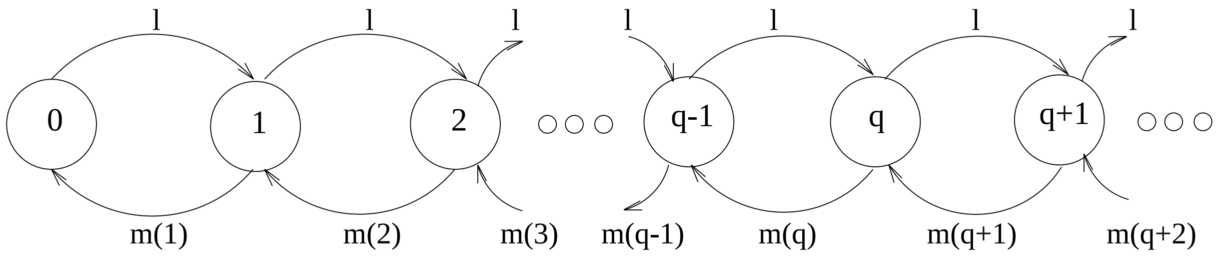


Figure 2.2: Markov chain for Ethernet

The above equation is based on renewal theory and holds for any distribution of Q and P. It shows that $E(Q)$ is sensitive to the packet size and the slot duration. Also since (2.1) converges rapidly to e^{-1} as $Q \to \infty$, $\lim_{Q\to\infty} W = e - 1$, and hence (2.3) yields an *asymptotic throughput efficiency* if we substitute the asymptotic value of W. If the packet size is the minimum feasible (i.e., $\frac{P}{C} = S$), (2.3) gives $E = e^{-1}$ showing that for small packet sizes Ethernet efficiency approaches the efficiency of a Slotted Aloha network according to their model.

A birth-death queuing model can be constructed from the above expressions as shown in Figure 2.2.

Each state q of the system represents the instantaneous load Q. Let the system be in state $q > 0$. It moves to the successor state $q + 1$ at a rate l (packets/sec) equal to the rate at which packets arrive at the system. Making the *infinite source* assumption this rate is independent of q. Adopting the simplifying approximation of *exponential service times*, we can consider that the system moves to the predecessor state $q - 1$ at the rate m_q (packets/sec) at which packets are delivered when the instantaneous load is q. This rate is a function of q and is given by the following equation which is the product of (2.3) and the carrying capacity of the network in packets per second:

$$m_q = E(q) \cdot \frac{C}{P} = \frac{1}{P/C + W(q) \cdot S} \tag{2.4}$$

In case that the variability of packet sizes is high, the above equation needs to be modified since high variable packet sizes violate the assumption of exponential service times, therefore their effect on the system performance is not negligible. State 0 corresponds to an idle system and packet transmissions cannot occur.

Solving the above model we obtain the probabilities $prob\{q\}$ of the system being in each state q. Then applying Little's law we can evaluate the average response time R as:

$$R = \frac{\sum_{q=0}^{\infty} prob\{q\} \cdot q}{l} \tag{2.5}$$

The numerator of (2.5) denotes the average population at the Ethernet subsystem. Almes and Lazowska [8] provide several response time vs. load $\equiv \rho = \frac{\lambda}{\mu} = \frac{l}{P} \cdot C$ curves proving that R depends on packet sizes. Moreover, this paper makes a comparison of the above model with simulation results concluding that the model has good accuracy.

Lam uses a single-server queuing model to approximate the distributed protocol of the Ethernet [9, 10, 11]. As such it is unable to capture many of the details of the protocol but it is computationally simple. It resembles the previous model in the sense that analogous assumptions are made (star topology, infinite population, time divided into slots) and similar results are derived. The model works with asymptotic values and yields an upper bound for the maximum attainable throughput (capacity). The slot duration is assumed to be equal to the round-trip propagation delay plus the jam time. The optimal retransmission algorithm is assumed such that the probability of a successful transmission in each slot is e^{-1}. This requires that full knowledge of the state of the system is available at all stations, and it is less optimistic than the Almes and Lazowska model, since Lam assumes $prob\{success\} > e^{-1}$ if $Q < \infty$. The mean number of slots from the end of the first collision until the next successful transmission is geometrically distributed with mean $e-1$. Stability is assured by the constant probability of a success. The model proves the sensitivity of the performance to the parameter $a = \frac{\tau}{P/C}$ which represents the propagation time normalized to the packet transmission time.

Tobagi and Hunt presented a detailed and more complex model for the CSMA/CD network for infinite and finite population systems [12].[2] The finite population analysis is much more complex than the infinite population one and in the limit the throughput predictions for both cases converge. Assumptions similar to the ones made in the above models are adopted here too. The paper also studied the effect of bimodal packet-size distributions on the system performance. It was shown that a small fraction of long packets is sufficient to recover a channel capacity close to the capacity achieved with only long packets. However the improvement due to the introduction of long packets is only experienced by the long packets. The throughput-delay performance of short packets is actually degraded, establishing the necessity to design and implement priority schemes.

Another modeling study for non-persistent CSMA/CD has been carried out by Coyle and Liu [14, 15]. The authors assume a finite number of hosts each generating new packets with an exponential distribution, which is more realistic than the total Poisson traffic assumption made by the previous study. They put emphasis on the retransmission strategies examining the stability of the system with respect to these strategies and the population of the system.

A throughput analysis of unslotted 1-persistent CSMA/CD is presented by Sohraby et al. [16]. This paper points out an erroneous methodology that had been adopted by Takagi and Kleinrock [17] and derives the correct throughput equation, showing that system throughput is maintained near capacity over a wide range of loads.

[2]This paper contains the analysis for non-persistent systems. The extension to 1-persistent case can be found in [13].

Furthermore Gonsalves and Tobagi [18] constructed a simulation model that dropped that balanced-star topology assumption. The authors examined Ethernet configurations where stations were uniformly spaced along the length of the network, organized in equal and unequal-sized clusters, and equidistant from a passive star coupler (star). The results showed that with stations uniformly distributed along a linear bus, "central" stations obtain better performance than stations at the ends. The effect of clustering the stations in equal-sized clusters was minimal. The introduction of inequalities in the cluster sizes increased the total throughput with larger clusters obtaining a greater than proportionate share at the expense of stations in the smaller clusters.

Finally Tasaka [19] examined the effect of buffering on the performance of slotted non-persistent CSMA/CD. This analysis can be extended for the 1-persistent case as well. The system under study is composed of a finite number of users, each having a buffer of finite capacity. The model investigates dependencies between the parameters of the system in order to derive the rescheduling probability and the buffer size that guarantee the system stability, keep the probability of buffer overflow sufficiently small and minimize the average packet delay.

2.1.2 Virtual Time CSMA: Description and Analysis

Virtual Time **CSMA** (**VT-CSMA**) introduced by M.L. Molle [5] is a new class of CSMA protocols with enhanced performance. Existing versions of the protocol include *slotted* and *unslotted* operation *with* or *without* the *collision detection* feature. Furthermore a very attractive extension of the algorithm to *multi-prioritized* environments is also available [6].

According to VT-CSMA, packets are scheduled for transmission by means of a 1-1 mapping that compresses the "real" axis (that contains the packet arrival times) until it fits onto a "virtual" time axis consisting of the union of all idle periods on the channel. The algorithm has two modes of operation: *"Backlogged"* (when transmissions have fallen behind arrivals because the channel has been busy) and *"caught up"* (otherwise). In the asynchronous mode (on which we are focusing) the mapping is implemented by the existence of an additional clock (called "virtual" clock) at every station whose operation is as follows.

As soon as the channel turns idle, the virtual clock starts running at rate $\eta > 1$ (note that the "real" clock advances at rate 1), in order to clear the backlog thus compressing the idle epochs of the real time axis by a factor of η. However if the virtual clock catches up with the real time, it slows down and keeps pace with the real clock (i.e., the virtual clock reading can never exceed real time, therefore satisfying the causality constraint). Every time the channel turns busy the virtual clock stops running (i.e. retains its current value) until the channel becomes idle again.

The rule for transmitting messages is now obvious. Each packet is "tagged" with its system arrival time (obtained from the real clock) and is transmitted when the station's

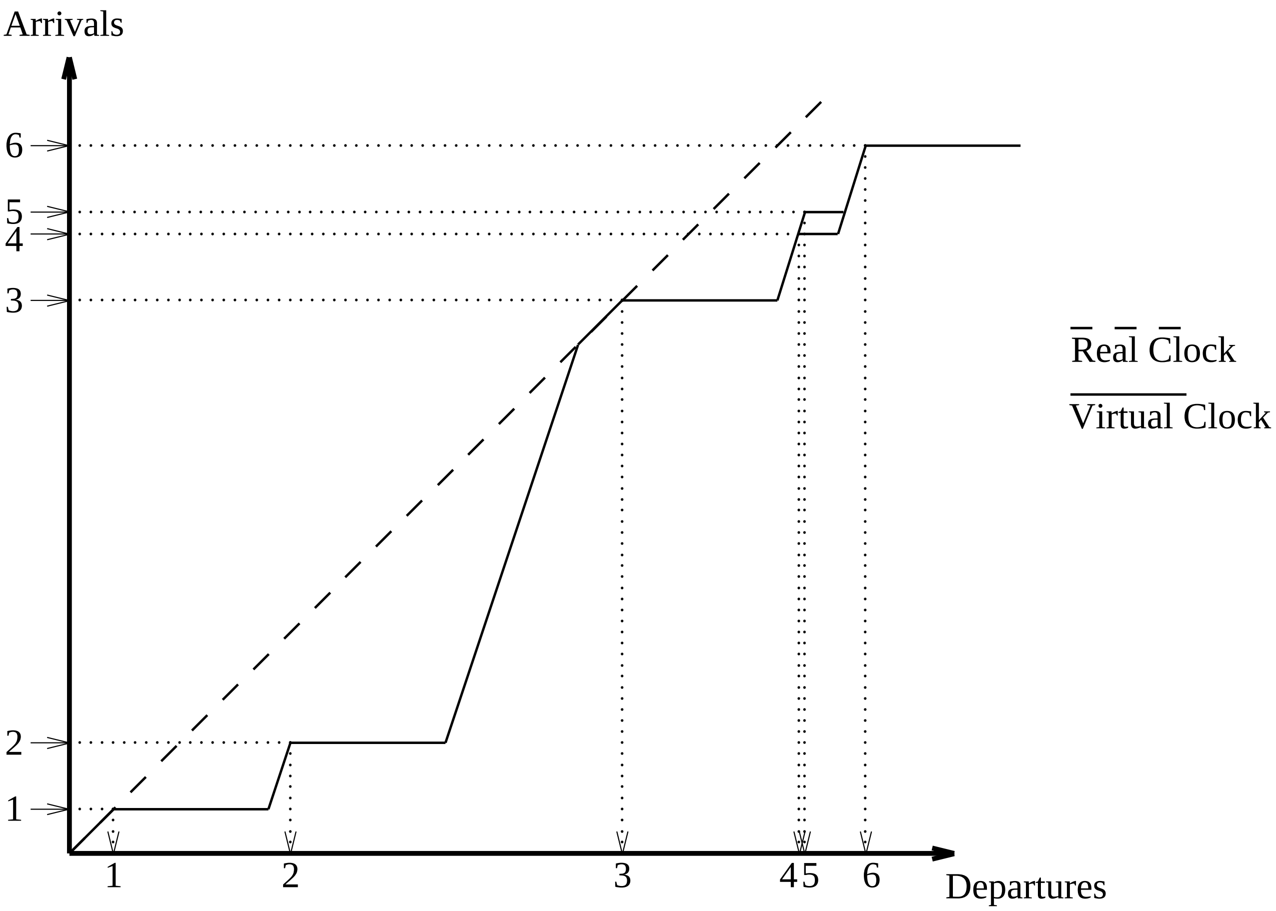


Figure 2.3: Mapping from arrivals to transmissions

virtual clock reaches or exceeds this value. In case of a collision, the "tag" value of a packet increases by the amount of the retransmission delay (derived by an appropriate retransmission feedback algorithm). At each station the packets are kept in a queue which is sorted by the "tag" value. Figure 2.3 shows the mapping from arrival times to transmission times. Messages 4 and 5 cause a collision. Since collision detection is available the transmission is stopped and rescheduled according to a backoff algorithm.

Molle has carried out the throughput analysis for all versions of the protocol under certain assumptions [5, 6]. The analysis shows that the capacity of the protocol depends on the virtual clock rate η. Moreover he derived a closed-form formula for evaluating the "optimal" value of the parameter η (termed η^*) for which the capacity of VT-CSMA approaches the capacity of the non-persistent CSMA protocol.

Estimation of delay follows the above analysis. The delay is the sum of the initial waiting time up to the first transmission, T_0, the delay at each unsuccessful attempt, T_R, and the packet transmission time and the propagation time during the final (successful) attempt. The method adopts the Poisson total traffic assumption to approximate the values of T_0 and T_R resulting in closed-form formulae for bounds for the delay. Those bounds appeared to be "weak" for the unslotted mode of operation.

The follow-up paper [6] extends the VT-CSMA analysis for systems in which messages of different priorities exist. In these cases each class of messages (i.e., those ones that have been assigned the same priority) has its own virtual clock rate which determines the priority of the class. The paper observes that the dominant term in the delay equation is T_0 because of the rare retransmissions. Based on this fact and incorporating analysis from the "moving server" queuing system [20], Molle develops a new approach to approximate T_0 and derives a closed-form expression for the delay, which gives remarkable agreement with the simulation results. The essential advantage of VT-CSMA is the simplicity and accuracy of its analytic model.

An implementation of VT-CSMA is presented by Konstantas [21] and Molle and Konstantas [22]. The system is represented as a collection of concurrent processes with each process reflecting the operation of a station. A detailed simulation model has been constructed for both single and multiple priority cases and a variety of retransmission algorithms have been tested. As a result of these tests, VT-CSMA seems to be insensitive to changes in the retransmission strategy. The result is of no surprise because VT-CSMA maximizes the probability of success in the first attempt, leaving little work to be done by the retransmission algorithm.

Finally Guardalben [23] discusses the performance of an heterogeneous CSMA network (i.e. one on which many CSMA protocols exist) and investigates circumstances where a combination of various CSMA protocols (namely 1-persistent, non-persistent and VT-CSMA) outperform each individual protocol. As it has been stated for the synchronous case [4], there is no protocol that can achieve higher capacity than VT-CSMA. This thesis focuses on the asynchronous versions of the above CSMA protocols and proves that in certain situations, coexistence of 1-persistent and VT-CSMA improves the overall throughput of the network. It is established that the "optimal" protocol is obtained by a combination of these two protocols only, and that the contribution of non-persistent stations would be of no help. The complementary nature of 1-persistent and VT-CSMA leads to optimality, although the definition of the optimal scheme is not trivial due to the many degrees of freedom of the problem. The thesis also deals with delay considerations of the mixed protocols.

2.2 Traffic Characteristics on an Ethernet LAN

We will now depart from the mathematical analysis of the CSMA protocols in order to present a characteristic set of papers related to measuring network performance on real

Ethernet configurations. This kind of approach is different and emphasizes measurements of network traffic, obtained by employing such techniques as *timestamping* via operating system utilities or monitoring tools.

The pioneering work in this area is due to Shoch and Hupp [24]. Their measurements took place on an "experimental" Ethernet (one that uses regular coaxial cable running at 2.94 Mbps) that spanned about 550 m and connected over 120 machines. The measurements pertain to network utilization, length distribution, source-destination pattern traffic and inter-packet arrival times. Daily observations showed that "typical" traffic utilized the network up to 37% for the busiest second. In most of the time the usage ranged between 0.60% and 0.84%. These results imply that computer communication applications tend to produce a "bursty" pattern of requests. The packets sent through the Ethernet system exhibited a bimodal distribution as far as their lengths are concerned. Most of the packets were "short" (terminal traffic, acknowledgments) although most of the total volume was carried in the "long" packets (file transfers).

Another interesting measurement is relevant to the *inter-packet times* which are defined to be times between the beginning of a packet's transmission and the end of the transmission of the previous packet. Their distribution comprises a number of "spikes" each of which corresponds to a "short" time slice and represents the percentage of times that the measured inter-packet time was bounded by the ends of the particular time slice. Several of these spikes represent traffic patterns due to request/response transactions with specialized network servers (such as file systems, printers, gateways etc.) or regularly repeated packet transfers according to a specific protocol. In a lightly loaded network (similar to the one that was being measured) packet exchanges due to communication between a host and a server are rarely disrupted by other packets. This reinforces the attribution of some of the distribution spikes to particular protocol operations. These observations reveal correlations between interpacket times. The cumulative distribution of inter-packet arrival times, which displays the probability of two successive packet arrivals within a time interval of specified length, shows that half of the observed packets are followed by another within 10 msec.

Finally, the authors examined the behavior of the network in an artificially generated heavy-load regime. This was achieved by presenting a sustained amount of traffic to the network. Ethernet showed no instability and its asymptotic performance (in terms of channel utilization) was in good agreement with the predictions of the simple analytic model described in section 2.1.1, for the various packet sizes that were tested.

The most elaborate measurement study of the Ethernet traffic on several diskless workstations environments has been conducted by Gusella [25, 26]. The 10 Mbps Ethernet under study connects 41 diskless workstations (of both SUN-2 and SUN-3 type) in clusters around 6 file servers running SUN UNIX operating system. The traffic volume observed here was 25 times larger than the one in the previous study. Network utilization examined more closely (in smaller intervals) proves the quite bursty behavior of network traffic which frequently reached levels over 30%. Such traffic would overload the Ethernet investigated in the previous measurement study. The packet length distribution is again bimodal but the average packet size is larger.

Gusella has examined network requests due to three different protocols that are running on the system, namely Network Disk (ND), Network File System (NFS) and Transmission Control Protocol (TCP). Measurements suggest that gateways would be potential bottlenecks since the largest percentage of the transferred data is associated with the ND and the NFS protocol and not the transport layer protocol (TCP) which supports inter-network traffic. Interarrival time[3] distribution (which is qualitatively similar to the one in the previous study) shows that there is high probability that one arrival would be followed by a second one within a deterministic time dependent on the protocol, the packet sizes and the traffic intensity. This is due to protocol features that cause clustering of fragmented large messages, thus resulting in generation of successive "arrivals" and "bursty" network behavior. Therefore the Poisson model that is widely used in the analytical studies is inappropriate to model these data, because it assumes independent arrivals.

In the follow up paper [27], Gusella attempts to characterize the burstiness of the packet arrival process by incorporating the use of *Indices of Dispersion.* These indices can reveal the correlation structure and the second order properties of a point process, such as the packet arrival process. There are many reasons for the high variability of this process. Protocol time constants, client-sever communication (involving remote procedure calls and interlock of send/receive operations), serialization of concurrent independent transmission streams, variable packet lengths, and sharing of file servers are responsible for correlations and dependencies in the arrival process. Gusella's measurements of source-destination traffic streams show that, although there is no noticeable correlation due to the last two issues, remote procedure calls, which impose a *synchronous* type of operation where a station sends a request to the server only after reception of the reply to its previous request, make substantial contributions to the dependencies in packet arrivals. Gusella defines two different types of indices of dispersion (namely Index of Dispersion for Intervals and Index of Dispersion for Counts) and discusses how standard analytical models such as *Batch Poisson Process* and *Markov Modulated Poisson Processes*, should be fitted to traffic measurements in order to take into account the variability of the data.

An accurate high time-resolution measurement study is described by Wilson and Leland [28]. Custom built monitoring hardware enables extremely complete and accurate

[3]Inter-arrival time is defined as the length of time between the beginning of the transmission of a given packet and the beginning of transmission of the previous one. Obviously, this implies that the packet transmission is included in the inter-arrival time which therefore introduces dependencies between inter-arrival times and packet transmission times in cases of networks allowing variable packet sizes. Note that *inter-packet times* have no correlation with the packet size distribution. Another interesting observation is that what Gusella defines as *inter-arrival* time, is actually *inter-departure* time. This is frequently misrepresented in the literature.

data to be collected, with tight bounds in the timestamp errors, overcoming discrepancies that occurred in similar previous studies due to loss of a small percentage of the LAN packets. The paper departs from Gusella's approach (which emphasized measurements of the send/receive streams of a workstation) and addresses the need for accurate characterization of the high packet rate aggregate LAN traffic. Leland's measurements pertain to the distributions of mean and peak packet arrival rates and interarrival times over a wide range of time scales (from hundreds of microseconds to weeks).

Peak rates verified the severe burstiness of the Ethernet especially at very low resolution. In order to provide a formal definition of "burstiness" for the complete range of time scales, Leland introduces the notion of index of dispersion[4] and plots the dispersion index I_t as a function of the length of the time interval t. The function is proven to be monotonically increasing which implies the unsuitability of the Poisson model to model the network traffic. An interesting observation that can be derived from this paper is that the index of dispersion for both internal and external traffic[5] has a great degree of similarity.

As far as the interarrival times are concerned, the paper stresses that their distribution is highly dependent on the particular protocol implementations and therefore simple distributions cannot capture the dependencies and interrelations due to high degree of traffic multiplexing.[6] Alternatively it proposes the use of Palm distributions for this reason.

Finally, one of the most widely referenced Ethernet papers in recent years is by Boggs et al. [29]. It presents the results of measurements of a stress-test on an Ethernet connecting several identical "Titan" RISC PC's manufactured by DEC. The hosts are attached to a 3000 ft. coaxial cable, by means of 4 regularly spaced DELNI multiport repeaters forming a bus network (see Figure 2.4).

The Ethernet controller of each station sends out a packet and then it must be serviced by interrupt code before it can send again. This interrupt latency (elapsed time between assertion of the interrupt and resumption of user code) is 100 μsec. Each host station has a flow of control mechanism that ensures that no more than 15 packets are buffered in it. Packets in the transmit queue of a station are addressed to the same station. Thus, this scheme is equivalent to each station having a circulating packet, because each packet is inserted in the transmit queue of the same station after its successful transmission from the particular station.

A number of tests have been carried out involving 9 different (constant) packet sizes and 8 different bimodal distributions of packet sizes on 3 different sized networks. During these

[4]This index corresponds to Gusella's *Index of Dispersion for Counts* and is defined as the ratio of the variance of the number of arrivals during a time interval of length t to the mean number of arrivals in t. The value of this index for the case of the exponential distribution is evidently 1.

[5]*External* traffic refers to traffic that is originated in a LAN but its destined to another, whereas *Internal* traffic comprises the entire traffic on the network. The ratio of the volume of external to internal traffic is typically 1:20

[6]This mainly applies to *broadband* networks which we will elaborate in the next chapter.

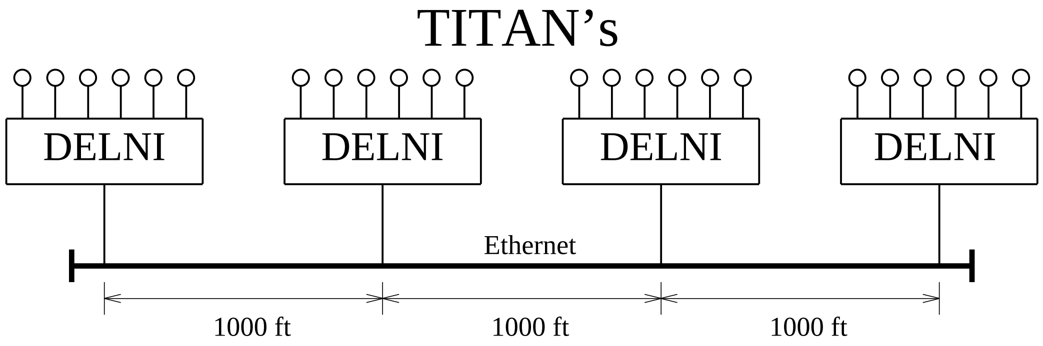


Figure 2.4: System measurement configuration

tests the authors varied the number of hosts from 1 to 24 trying to balance the number of hosts per repeater. Each test consists of a number of test intervals. Each station collects statistics in the middle of each interval so as to capture steady-state system behavior. At the end of the test it sends the accumulated data to a controller station which does the necessary calculations extracting the aggregate statistics.

The results showed that Ethernet is capable of good performance for a typical application even at high offered loads. They suggested that the network can never be pushed to instability as a result of high load of useful traffic. On the contrary, the only cause that could possibly lead to saturation would be hardware failures or software design bugs generating accidental high loads. Also use of a linear backoff algorithm may force unstable behavior especially at high loads. The authors recommended to keep the network size short, use large packet size and not to mix heterogeneous workload applications in order to assure good Ethernet performance.

2.3 Elements of Queuing Network Modeling

Queuing Network Modeling is a rapidly advancing discipline providing primary modeling tools for quantitative performance analysis of computer systems. It deals with a subset of general networks of queues, namely *separable* queuing networks along with several extensions that are necessary to accurately represent particular system characteristics. Restriction to this subset is mandatory in order to provide efficient algorithms for obtaining performance measures of realistic computer systems, and implies a number of assumptions that ought to be satisfied by the system under study. Using these algorithms the systems analyst can predict the behavior of the system (and possible variations of it) at low cost, as opposed to implementing and assessing each system modification at the relatively higher expense of simulation.

The key property of the separable networks is that their *state distribution function* has a form of a product of factors, each of which is associated with a queuing center in the model [30]. Hence, the components of such a system are "separated". The separability leads to simplifications in the mathematical description of the behavior of the system and can be outlined as follows [31, 32, 33]:

- *Job Flow Balance:* The number of arrivals at each center is equal to the number of completions there.
- *One Step Behavior:* The system changes state only through the movement of a single customer from one service center to another.
- *Routing Homogeneity:* The proportion of customers routed from station i to station j following completion of service at station i is independent of the state of the system.
- *Device Homogeneity:* The time taken to service a customer at a given center depends only on the number of customers present in its queue.

There is a large number of networks that violate one or more of the above separability constraints (non-separable models), for which we can apply either *global balance* or *Hybrid Modeling* techniques at the high cost of massive amount of computation, or use modifications of the efficient algorithms that result in approximate analytical solutions [34].

In **Q**ueuing **N**etwork **M**odels (**QNM**s) the modeled system is represented as a collection of *service centers*, which normally correspond to hardware resources and their software queues (if necessary), and a number of *customers*, which represent users or transactions. Model definition consists of the description of the above two entities [33]. Service centers may be either *queuing* or *delay*. Customer description pertains to workload intensity (offered load) definition. This can be done in three ways, each of which is characteristic to a specific type of workload:

1. By specifying the rate that customers arrive to the system (*transaction workload*).
2. By specifying the average number of active jobs, which is often called Multiprogramming level or MPL (*batch workload*).
3. By specifying the average number of active terminals (users) and the average time between interactions submitted by a user, often called "think time" (*terminal workload*).

If all customers in the system are assumed to be indistinguishable, we have a *single-class* model; otherwise we must differentiate customers with various characteristics and must use a *multiple-class* model. Usually today's computer systems are represented by multiple-class models because of the diversity of applications they support and the differences in the

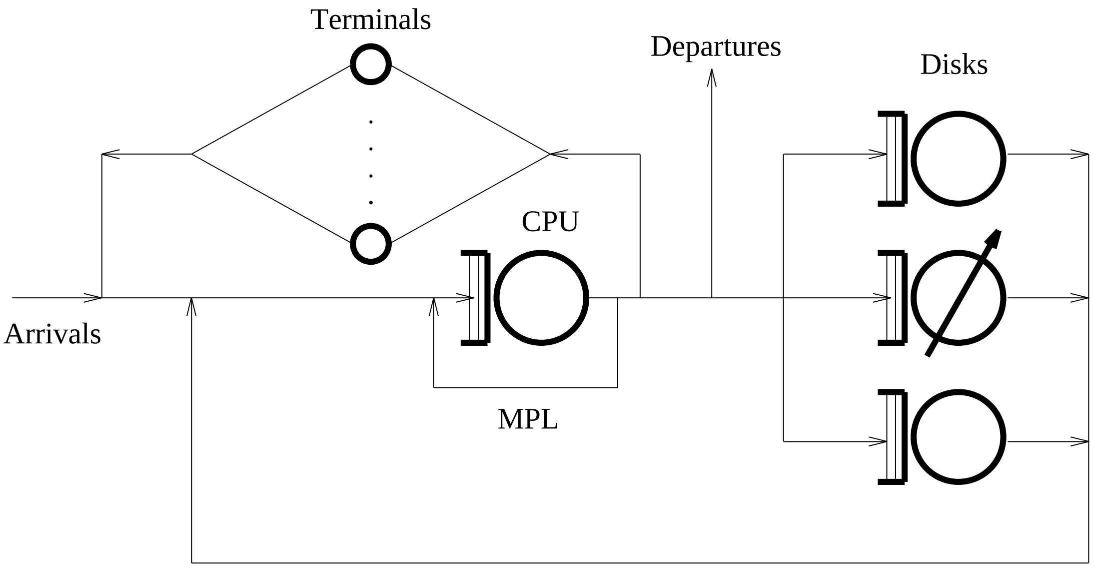


Figure 2.5: A Model with mixed workload

nature, the characteristics and the demands that dominate each application. In a multiple-class model, the scheduling discipline at the service centers can be important because customers are distinguishable. Disciplines that can be represented in **QNMs** are *First Come First Served* (FCFS), *Processor Sharing* (PS), *Last Come First Served-Preemptive Resume* (LCFS) and *Infinite Servers* (IS) [33, 35, 36].

Models with batch or terminal workloads (where the number of customers in the system is fixed) are referred to as *closed* models, whereas networks with transaction workloads are characterized as *open* models. Models consisting of both open and closed types of classes are called *mixed* models. Figure 2.5 illustrates such a model.

The description of the flow of customers in such a *network of queues* (i.e., which queue will a customer visit next, after it has completed service at a given queue) is termed a *routing chain*. The description is accomplished via a transition probability matrix, often called a *routing table*, that shows where a customer will go next after visiting a particular queue. Another way to describe the routing of customers in the network is by using the *visit count* of the service centers which is defined as the ratio of the number of completions at a resource (center) to the number of system completions. Single-class models are said to consist of a single routing chain, in contrast to multiple-class models that contain multiple routing chains [35]. However in several cases we may have multi-class single routing chains where customers change their classes as they circulate in the network.

The process of *model parameterization* accomplishes the definition phase. Here we measure the workload of the system of interest and translate it into inputs of a *QNM*. For single-class models we need to provide the total amount of time that a single customer requires in service at each center of the system on average (*service demands*). Knowledge of either the routing matrix or the visit counts, and the *service times* (time that a single visit requires at each center), suffices to determine the required service demands. For multiple-class models we must supply the demands of customers of each individual class at every center. The parameterization process in very crucial due to the sensitivity of the model outputs to the inputs. The accuracy of the inputs determines the reliability of the model and its predictions and their correct definition becomes a very tedious and difficult job.

Finally we can evaluate the QNM, calculating the values of the performance measures by means of efficient computational algorithms. Typical model outputs are *throughput* and *response time* on center or aggregate basis, and *device utilizations* and *queue lengths*. For multiple class models, these measures are also provided on a per-class basis. The solution of the model must be validated by comparing the model outputs with the statistics obtained by extensive system measurement and benchmarking. A validated model can afterwards be used to predict the effect of system modification or upgrade with a great degree of confidence.

The solution to a separable QNM has a standard form, that is easy to compute up to a normalization constant. Buzen [30] provided the first method for efficient model solution by evaluation of the normalization constant. His method (*convolution algorithm*) is based on a recursive definition of this constant. The computation can be done in time proportional to the number of centers and the number of customers (for a single-class model). Performance measures are generated via manipulations of the normalization constant. One drawback of this algorithm is that numerical overflow/underflow problems may occur during the computation of the normalization constant.

Another algorithm with asymptotically equal time complexity is *Mean-Value Analysis* [37]. The MVA algorithm is easier to implement and does not suffer stability problems. The algorithm has the advantage of directly computing all the performance measures for each population in the queuing network. However, it has the disadvantage of computing only average values, so it is impossible to construct the complete steady-state probability density directly from the results of the computation. The MVA algorithm is based on the arrival-instant theorem [37, 38], which states that a customer arriving to a queue in a product-form queuing network sees precisely the same distribution of customers as an outside observer would see in steady state, if one less customer was circulating in the network. Besides that, it uses Little's law, applied to each routing chain and separately to each service center, to furnish a set of equations which are easily solved numerically.

The "exact" version of the algorithm is based on a recursive relationship over the population of each closed routing chain. Therefore the cost of computation can become prohibitive for many computer modeling applications requiring many different customer classes. For this reason a number of heuristic extensions of the algorithm have been devised. Those extensions (referred to as approximate MVA or aMVA algorithms) replace the recursion over all possible population levels by an iteration for fixed populations thus resulting in approximate solutions.

Amongst the techniques that are used for the approximate solution of non-separable models is *Hierarchical Modeling.* It has gained a lot of popularity due to its simplicity and is often applied to modeling Distributed Systems and Local Area Networks. It is based on the use of *Flow Equivalent Service Centers* (FESCs) which can be used to represent the behavior of entire subsystems. These centers are "load-dependent" meaning that their service rates are functions of the number of customers residing in their queues.

Systems with FESCs can be evaluated by using an extension of the original MVA algorithm, so as to compute the probability density function of the number of customers in the queue. These extra calculations must be done at each iteration (corresponding to each population level) in order to estimate the average number of customers in the queue and the average residence time. Using the arrival instant theorem we can compute the probability distribution for the next population level. The price that must be paid is higher computational complexity (the relation now becomes quadratic with respect to the number of customers).

There exist several commercial performance evaluation software packages that have implemented many of the above techniques. Such utilities free the analyst from the algorithmic portion of the modeling process and allow him to concentrate on the other important issues. One package that uses the aMVA approach and we have available is **M**odeling and **A**nalysis **P**ackage (**MAP**) [39, 40].

2.4 Models of Distributed Systems

Recently we have witnessed a proliferation of distributed computing environments. This is a result of technological advances in the area of hardware device manufacturing, reducing the cost of computers, along with software advances allowing the development of communication protocols with high performance and reliability. Many of these environments are built around Ethernet interconnection. The evolution of these distributed systems has affected the way in which the Ethernet network is currently being used. Initial applications involved *remote terminal access* and *file transfers.* Later ones dealt with *diskless workstations* and *distributed processing.* The current trend is *client/server* architectures, where the Ethernet is used to provide distributed access to specialized services provided by file servers, print servers and disk servers. There exist several studies that have focused on Ethernet performance in many of the above environments. In the following we will illustrate the key points that can be derived from those studies.

Lazowska et al. [41] were among the first to investigate the performance of single-user workstations that access files remotely over Ethernet. The authors measured elapsed times and service demands for local and remote file accesses on a variety of different system configurations. The "canonical" system that was studied is shown in Figure 2.6. The workstations are SUN 2's, running the UNIX 4.2 BSD operating system, and the network is a 10 Mbps Ethernet.

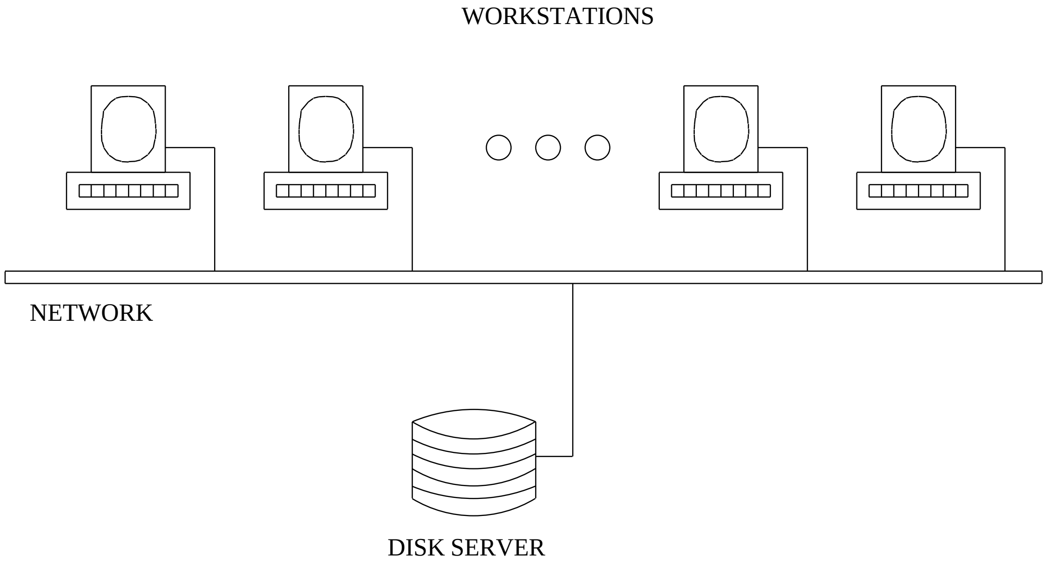


Figure 2.6: A diskless workstations system

The measurements were used to parameterize QNMs for the performance evaluation of that system. Once the outputs of the baseline model were verified against the real measurements, the model could be used for a better understanding of the issues arising in the design of diskless workstations and their file servers, for designing alternative configurations and for modification analysis.

It is obvious that the "accurate" model for this system is a non-trivial one because it is not easy, and in some cases not possible, to represent the system characteristics in sufficient detail via use of a separable QNM. There are certain factors that either do not meet the separability constraints or are very tricky to model, namely *simultaneous resource possession*, *Direct Memory Access*, *overlapped activity*, *paging*, *swapping*, *buffering* and *blocking*. Although the model used is not described in the publication, we can provide the service center description of it. Each workstation is represented by a queuing center, the file server by two queuing centers (one for the Server CPU and one for the Disk unit), and the Ethernet by a Load-Dependent service center. The network representation in such a way is imposed by the fact that Ethernet efficiency depends on the number of workstations simultaneously attempting to assert data. The service demand at the Ethernet center is given by the denominator of (2.4).

Although the emphasis in the above research was the overall system performance, what is important to our study is the network behavior. The paper concludes that a single

Ethernet is capable of handling efficiently a large number of file servers and their associated workstations. It claims that network contention was not a serious problem in those experiments. Nevertheless, we should point out that the hardware on which that study was carried out is considered to be obsolete nowadays. The increased speed of today's contemporary systems has dropped the ratio of Service demand at the CPUs to the service demand at the network, and has enabled the execution of more operations at the CPUs yielding higher network traffic and contention. So the question of satisfactory network performance becomes vital.

An interesting issue in the above paper is the surprisingly low paging ratio, which was measured to be only 25% of the whole file access activity. This observation led to the conclusion that replacement of the local (paging) disk would not have a profound effect on the network traffic. The above ratio – although it depends on the nature of the running applications, does not seem to be reasonable, especially for workstations equipped with 2 Mbytes of main memory. In today's systems, the level of paging activity is remarkably high [26, 42]. However, the existence of a local paging disk at each workstation, limits the amount of swapping and alleviates the network congestion.

Another study that involves a more modern architecture is presented by Molloy [42]. This paper focuses on the validation of an MVA QNM for a client/server oriented configuration. The system was composed of Hewlett-Packard (HP 9000) workstations and disks. Molloy used input from measurements of a particular workload on a single client workstation with another station acting as the server, in order to parameterize a QNM and obtain the performance measures. The architecture under study was a cluster of client workstations served by another workstation with a disk via use of an Ethernet local network. The issue is performance vs. load (i.e number of clients). This client/server pattern is particularly suited to systems based on diskless technology since in such a distributed computing environment memory contention is independent of the number of clients. Additional users contend only for the disk server. Conversely, in other type of architectures (e.g. Time Sharing Systems, X-windows) memory contention (and therefore increased paging activity) becomes a serious issue.

The model used here is a closed multi-class model. A customer may change its class type as it circulates in the system. There are three classes that represent the customers that are in the *"think"* mode, the ones that execute a *local* transaction and the ones that execute a *remote* transaction respectively. Molloy investigated workloads created by three different applications separately. The result revealed that network utilization remained below 25% and that the disk could accommodate between 8-15 clients (depending on the application) before it saturated.

However, it would be useful to stress that in the above applications, a large percentage (over 50%) of the transactions were local and did not cause any network traffic. Since the disk was the bottleneck device in the above system, it would be desirable to use a larger number of server workstations and disks in order to handle the workload created by the clients efficiently. This would have an immediate effect in the network traffic and could possibly have led to network saturation. In the above perspective, Molloy's experiments are not sufficient to show the system and particularly the Ethernet behavior in an "enlarged" configuration. Therefore additional tests must be performed in order to figure out the impact of a more network-intensive workload to the system behavior.

A more detailed study that examines the Ethernet performance on the new network-oriented windowing system termed *X-windows* and particularly its DEC implementation ("DECwindows") is illustrated by Mirchandani and Biswas [43]. The operation of the X-windows is based on a client/server model. The display workstation (terminal), which is responsible for providing graphics services, is the *server process*, whereas application programs – which are typically represented by windows on the display, are the *client processes*. Clients can be *local* if they execute on the same workstation as the server, or *remote* if they run on a different computer. Client/server communication is achieved through the X-protocol over a *connection*. Typically, multiple clients have connections to the same server simultaneously.

In this particular DEC distributed computing environment, local clients communicate with the X-server via shared memory, whereas the connection between remote clients and the X-server is implemented via X-protocol messages that are sent according to "DECnet" (which is DEC's transport layer protocol). The distributed file system under consideration (which is called "VAXcluster") is a disk block access mechanism and is based on a *data sharing model*.[7] In order to synchronize accesses to system resources, VAXcluster uses *remote lock request* messages, which operate in a synchronous request/response fashion, thus managing a session and facilitating data transfer.

The question that has been addressed in this article is whether the Ethernet would be a bottleneck in the above environment, especially when hundreds of workstations simultaneously ran DECwindows applications remotely over the Ethernet. The test configuration consist of two VAXstation 2000s, which are connected over an Ethernet segment. One of this stations is set up as disk server, whereas the other one is used to run the X-server. Most of the clients are remote, executing in the disk server workstation. The modeling technique that was used here is based on the parameterization of a simulation model via use of traces from the above two station system.[8]

The authors tried to capture the network traffic and translate that to packet size and inter-arrival time distributions, in order to pipe them into their simulation model to obtain the required performance measures. The collection of the network traffic information was not done by a LAN analyzer. As we mentioned before, there are three sources of Ethernet traffic in the particular computing environment: X-protocol messages, remote

[7]This implies a low-level operation, as opposed to SUN's NFS file transfer protocol, which provides *file* and not *block* accesses, leaving locking functions to be implemented by lower level protocols.

[8]This resembles Molloy's method in the previous paper. However, he used system measurements to parameterize an MVA-based model and not a simulation one.

disk accesses and lock messages. In the first stage, each of these components was traced separately (by activation of an appropriate collection tool). Then these traces were post-processed to extract the relevant to the experiment information and were given as input to an emulation program. That program emulates the VAXcluster and DECnet protocols and does the necessary transformation of the above workload into inter-arrival time and packet size distributions that would be seen on the Ethernet. A network analyzer was used for validation of the protocol emulation. The argument used was that complete network characterization would not be so effective if it is done by a LAN analyzer since most of the analyzers observe only a portion of the traffic and therefore have limited precision. The authors claim that the above method can be extended in order to be applied to other similar non-DEC environments.

The applied workload was a user-interaction intensive one, reflecting a worse-case scenario. A variety of cases were examined, including different packet sizes and different values for the parameters of the previously mentioned distributions. The result was that performance degradation due to network saturation (particularly saturation of the Ethernet adapter of each station) depends on the specific application, as well as on the network traffic parameters (such as inter-arrival time and packet length distributions). The network utilization leveled off between 50-400 workstations proving that the Ethernet may saturate early if a network intensive application is considered. In most cases, however, the limiting configurations were in the region above 150 workstations, showing that Ethernet can support relatively large configurations with acceptable performance.

The above methodology was application and system dependent. It assumed availability of a variety of measurement and monitoring tools and appropriate software packages. Therefore it cannot be easily applied without the existence of the necessary equipment. Furthermore the information about the system, and especially the networking part, that are provided are insufficient so as to enable a modeling study using an MVA type of approach. However, the above technique and the accuracy of the results reveals the possibility of performance degradation as an effect of network contention.

We should, however, point out the low speed of the workstations on which this particular experiment was conducted. These stations are not even rated as MIPS! Current state-of-the-art DEC workstations (e.g DECstation 3100 or DECstation 5100) are at least 12 to 25 times faster than the VAX 2000s, supporting substantially increased packet transmission rates and fast communication protocols. Therefore the above results should be considered very optimistic in light of today's applications.

Chapter 3

Ethernet Simulation Model

Ethernet uses an unslotted distributed access mechanism, which has proven to be difficult to model analytically, mainly because of the employed retransmission backoff algorithm. Thus, the best way to carry out a detailed study of its characteristics and accurately assess its behavior is with a simulation program. Even a simplified model, adopting a few assumptions to ensure run-time efficiency, can prove that in certain cases, current analytical studies of the Ethernet are not capable of providing satisfactorily accurate results.

In our case we have developed an Ethernet simulation model that accurately follows the details of the protocol on the equi-distant "star" topology. Our simulation has the additional ability of representing closed systems that can be easily modeled as closed Queueing Networks. These systems have a straightforward analytic solution by means of MVA techniques. Documentation of the model specifications and implementations details is the subject of this chapter.

3.1 System Specifications

Ethernet has been a very successful architecture and numerous implementations of it have been reported in the literature. Understanding of the fundamental high-level implementation aspects is necessary for defining the specifications of the system that we want to represent. These specifications are then used to construct the simulation model. In this section, we survey information relevant to the Ethernet technology and set the parameters of our experimental system.

3.1.1 Typical Network Implementation

Xerox Corporation began developing a LAN system known as *experimental Ethernet* in the early 70's. The design was very successful and its use soon became widespread. Almost at the end of the decade, several companies (DEC, Intel, Xerox) jointly defined an Ethernet specification for standardization purposes [44]. The early Ethernet specifications made substantial contributions to the standardization effort done by the IEEE subcommittee 802.3 [7, 45]. The 802.3 subcommittee was established in 1980 (as a part of the IEEE 802 project) in order to provide internationally accepted standards for the CSMA/CD access control method, enabling the intercommunication of equipment from a variety of manufacturers. The work of the committee came to fruition in 1985 and as a result the protocol definition ensured compatibility to the one that was announced by the above three companies.[1]

The above IEEE work, addresses only the lowest two layers of the Open Systems Interconnection (OSI) reference model [46], by defining standards that must be obeyed by all Ethernet manufacturers. Functions that have been documented for the first layer of the model (*Physical Layer*) are: *Data Encoding/Decoding* (which performs bit manipulation and station synchronization, according to Manchester encoding/decoding scheme), and *Physical Medium Attachment* (which performs bit transmission and reception as well as carrier sensing). Respective functions for the second layer (*Data Link Layer*) are: *Data Encapsulation/Decapsulation* and *Link Management*. The first one deals with the construction of transmission frames from bits, which are actually the data units that are transmitted on the passive broadcast medium. The second one deals with *collision avoidance* (i.e., monitoring of the carrier sense signal and deferring transmission policy), and *collision handling* (i.e., transmission of jam signal and scheduling of retransmission attempt).[2]

The frames formatted by the data encapsulation function contain additional information fields in the beginning and at the end of the useful data part. This information is used for purposes of receiver synchronization, frame delimiting, sender and receiver station identification and transmission errors detection. Such additional fields are: *Preamble* (8 bytes), *Station Addresses* (12 bytes), *Packet type* (2 bytes) and *Frame Check Sequence* (4 bytes). Ethernet frames are of variable length, with no frame larger than 1526 *octets* (bytes) and smaller than 64 octets.[3]

According to the Data Link Layer specifications "back-to-back" (consecutive) frames must be separated by an interframe space of at least 9.6 microseconds. However analytic studies have shown that in certain cases (especially in bulk file transfers) imposing a larger interframe gap between frames transmitted from the same station could lead to better

[1] IEEE adopted the initial Ethernet specifications (with minor changes) as the IEEE 802.3 10Base5 (10 Mbps, Baseband, 500 meters per segment) standard.

[2] In fact, the above two functions belong to the *Media Access Control* sublayer which is the bottom sublayer of the Data Link Layer. The upper sublayer (*Logical Link Control*) does not differentiate media access control protocols. IEEE 802 has defined a single standard for this sublayer, which applies to all protocols and deals with data flow organization, command interpretation, error control and recovery functions etc.

[3] Restriction in the minimum packet size is imposed by the maximum allowable slot time for the backoff algorithm. This maximum slot length consists of the maximum round trip propagation time (46.4 μsec) and the maximum jam time (4.8 μsec), which is equivalent to a 64-byte packet transmission time.

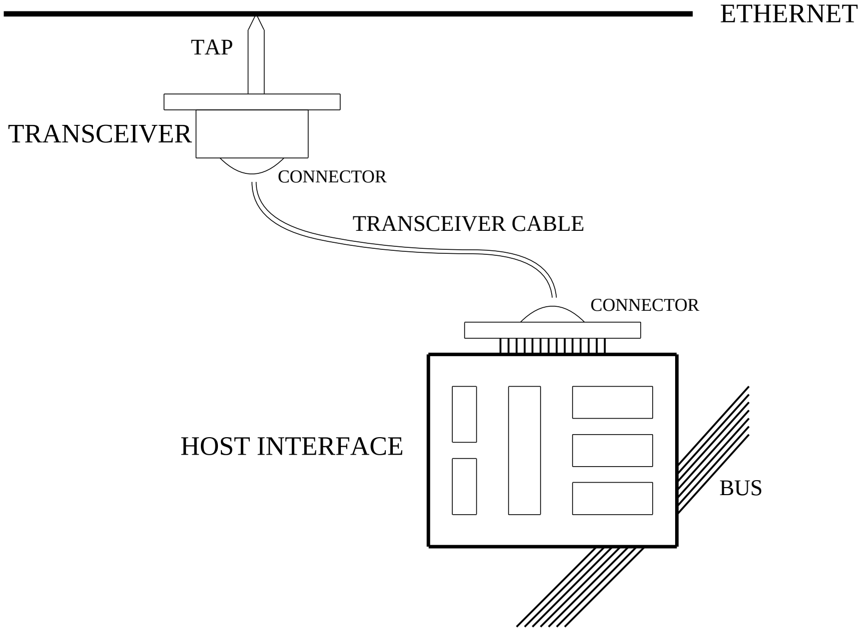


Figure 3.1: Ethernet connection block diagram

performance [47].

Figure 3.1 illustrates the physical implementation model [48, 49]. Each connection to the Ethernet has two major electronic components. The first one is a device called *transceiver* which is tapped to the Ethernet cable[4], sensing and sending signals on it. It implements the channel access function and is located close to, or directly on, the coaxial cable. The second one is the *host interface* which is mainly a controller board that connects to the transceiver via a *transceiver cable* thus allowing the host to be a short distance away from the Ethernet cable. This ensures network reliability since failure within a node (which is isolated from the channel) will not affect the rest of the network. The transceiver cable carries power to operate the transceiver as well as controlling signals, and it can be up to 50 meters in length. The host interface also communicates with the computer, usually through the computer backplane bus. Typically, the host interface includes a *serial interface* (to communicate with the transceiver) and a *data link controller* (a dedicated co-processor). It performs a variety of functions such as frame manipulation, address filtering (keeping unwanted frames from passing to the host), transmission errors detection and network link management. While the transceiver is a simple hardware device, the host interface is usually a complex hardware device using a microprocessor for traffic control.

[4]The Ethernet channel is typically a low noise shielded 50Ω coaxial cable over which, information is transmitted at 10 million bits per second.

The popularity of Ethernet, which assures a high-volume market, has led several manufacturers (Intel, AMD, National) to commit the necessary resources in order to develop VLSI chips implementing the Ethernet functions. Intel [50] provides a series of 3 chips that implement Ethernet. The first one is the 82C502 Ethernet Transceiver chip whose primary functions are data transmission/reception and collision detection. The second one is the 82501 Ethernet Serial Interface which performs Manchester encoding/decoding, drives the transceiver cable, and communicates with the third chip (82586) which is the Intel's LAN coprocessor. As such, 82586 does not depend on the host CPU for time-critical functions (e.g. frame transmission and reception). It can be programmed to support a variety of specific applications and variations of the CSMA/CD protocol, and interface with a large number of microprocessors. It achieves efficient use of memory by saving the large amount of short frames in buffers of minimal size, and storing long frames by successively chaining buffers together. Finally, it provides a big set of network management and maintenance capabilities as well as Direct Memory Access (DMA). It is interesting to point out that the above Intel's implementation of Ethernet does not allow continuous transmission of packets by a station, via a use of a *"watchdog" timer* which has been integrated into both 82501 and 82C502 chips. Therefore generation of back-to-back packet cannot be done.

Although Intel chips have been very popular, they are gradually giving way to Ethernet chips manufactured by Advanced Micro Devices (AMD). AMD's solution consists of a set of chips, namely the Am7990 Local Area Network Controller (LANCE), the Am7992B Serial Interface Adapter (SIA), the Am7996 Transceiver, and the Am79C900 Integrated Local Area Communications Controller (ILACC). The Am7990 is designed to operate in conjunction with the Am7992B and the Am7996 chips (which implement the Serial Interface and transceiver functions respectively), within an environment that includes close coupling with a local memory and a microprocessor It has programming capabilities via appropriate registers and built-in data structures.It performs on-board DMA, advanced buffer management and extensive error report and diagnostics. This hi-tech chip permits transmission and reception of back-to-back packets with interframe gap time of as low as 4.1 μsec. The last chip (Am79C900 ILACC) is a second generation Ethernet controller that offers a migration path to 32-bit applications. It has higher level of integration and performance, and assures compatibility to the software written for Am7990 LANCE. Many other computer manufacturers (e.g. DEC) are using the above chips to construct Ethernet controller boards [51].

Ethernet networks ordinarily use *baseband* signaling, which is a digital form of transmission. A problem that is usually encountered is the distortion of the signals as they travel along the cable. This difficulty can be overcome by using *repeaters* which receive bit level digital signals and retransmit them at their original strength and sharpness. In this sense the cable is divided into *segments* that are connected by repeaters. Specifications restrict the length of a segment to a maximum of 500 meters. At each end of a cable segment a *terminator* is attached. This passive device provides the proper electrical termination to eliminate reflections. Repeaters use node positions and contribute to the total node count on both segments that they connect. Only two repeaters may be placed between any two machines,[5] so the total length of a single Ethernet is still rather short (1500 m).[6] A typical Ethernet layout consists of a *backbone* segment that runs vertically along the floors of a building, and at each floor connects to an Ethernet segment that spans the particular floor. In the above network repeaters are acting like *"gateways"* that interconnect individual network segments to form a larger extended network.

The initial Ethernet implementation used a relatively expensive coaxial cable, which is now referred to as *thick Ethernet cable*. When the cost factor becomes important, a standard coaxial cable (like that used for cable television) can be used to implement Ethernet. Called *thin Ethernet cable* this thin-wire cable is inexpensive but supports fewer connections and is limited to shorter distances. Thin-wire networks achieve further cost reductions by the use of less complicated hardware, for instance, digital network interfaces that eliminate the need for transceivers.[7]

In several cases cost can also be reduced by incorporating the *broadband* analog transmission technology which enables multiplexing, several Ethernets onto a single cable instead of using many cables. Distinct advantages of broadband technology are: inexpensive and user-installable cables and connectors as well as greater bandwidth utilization.[8]

Most recently, a new form of Ethernet known as 10BaseT, which runs on ordinary twisted-wire-pair telephone wiring has been introduced [52]. This sophisticated approach will alleviate problems occurring during the network installation phase and give additional flexibility to the network design. This 10 Mbps baseband network that transmits over twisted pair cabling can be considered to have a *star* rather than a *bus* configuration. Instead of having a coaxial cable line that connects individual stations, here each workstation is connected directly to a *hub* (usually a repeater) which retransmits a signal to other points on the network.[9] Hence, we can discern a trend towards the use of star networks that incorporate innovative twisted-pair technology (such as 10BaseT), unlike the original Ethernet design that assumed a linear or unrooted tree topology. Although this

[5]The limit to the number of repeaters is because they gobble up preamble bits.

[6]However several manufacturers (DEC) allow use of remote repeaters (which are connected by fiber cable up to 1 km long), thus extending the network length to a maximum of 2.8 Km. This also includes the length of 6 transceiver cables.

[7]These features resemble the 10Base2 IEEE standard, which is also known as *Cheapernet*. This limits the segment length to 185 meters and the number of nodes per segment to 30. The transceiver function has also been integrated into the node. The Intel and AMD chips that we mentioned before support Cheapernet too.

[8]Such a broadband-based architecture was the IBM PC Network which is supported by the IEEE 802.3 standards.

[9]This appears to be an upgraded version of the older IEEE STARLAN standard, which operated at 1Mbps twisted pair telephone wiring. This cabling scheme involved easy installation, reconfiguration and servicing.

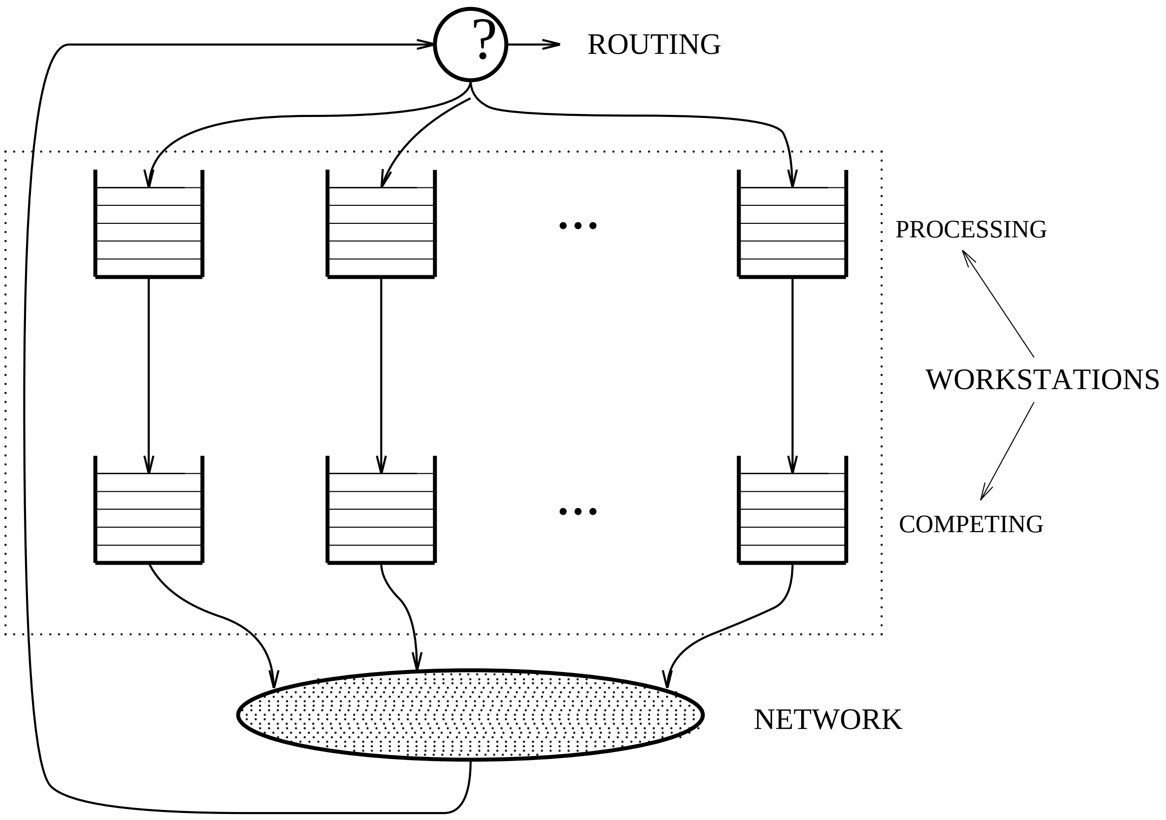


Figure 3.2: Simulated system structure

kind of network topology is probably bad for performance [53], its symmetry alleviates the complexity of modeling.

3.1.2 System Components and their Operation

We will now present the characteristics of the system that we have simulated for the purposes of this report. The novelty of this simulation model is its ability to represent separable closed queuing networks that are easy to solve by means of an MVA algorithm. That is, unlike previous models, where the system under study was basically an open system with external requests arriving according to a certain distribution (usually Poisson) and departing from it after having received service (e.g. transmission over the network), the emphasis in our model is *closed* systems where there is a constant number of "circulating" requests.

A *request* is defined to be a frame transfer over Ethernet accompanied by a finite amount of processing. The population of requests is predefined and remains unchanged throughout the duration of the simulation. This eliminates the need for assumptions often made in theoretical studies, namely inter-arrival time distribution and buffering at the workstations. In this system, a request, upon completion of service, is re-inserted to the system. Moreover, queue lengths cannot grow infinitely and therefore there is no need to impose limits on their size. This simulation program can also be used to model open systems with Poisson arrivals and –most important– can be extended to mimic a client/server configuration based on parameters obtained by published measurements.

The major system components are the *workstations* and the *network*. The workstations are responsible for providing the necessary "processing" to the data units and for organizing them into frames for transmission over the network. The amount of processing for each frame is constant,[10] and it is the criterion according to which stations are distinguished from one another. If the service time at the processing centers is the same, then all workstations are assumed to be identical. By altering processing times, we can reinforce distinctions on the "type" of the workstations. The option of no processing time is also possible and can be used to model cases where the network demand substantially exceeds the processing latency. Although this is not very likely to happen in real systems, it can be of theoretical interest in "stress testing" Ethernet.

A block diagram of the structure of a closed system is illustrated in Figure 3.2. This scheme does not represent the physical topology of the network but sketches the logical orderings among the operations that are taking place. The requests are first queued up for processing, which implies delays due to congestion at the processing centers. The typical implementation of this step is by ordinary FIFO queues. After completion of service, the requests are again lined up for service at the network center. Since we may have multiple requests at one workstation, we need to implement queueing centers again. The requests at the head of each queue are competing for acquisition of the network and "service" ends when they have completed their transmission. In case of collisions, the colliding requests defer their transmission and re-attempt after a finite period of time, determined by the truncated binary exponential backoff algorithm. In the meantime other requests that are queued behind the ones that are backing off, must wait before they attempt transmission. At this point the details of the actual LAN protocol are coming into light. These will be further analyzed later in this chapter.

The *routing policy* is a mechanism that determines the destination of a request after its successful transmission. In real Ethernets this is done deterministically by explicit definition of the destination address. However for the purposes of a closed queuing network we want to be able to use several randomization schemes such as *uniform routing* where

[10]The assumption of deterministic processing times at the workstations, violates the assumption that the service times at FCFS queuing centers must be exponentially distributed, for product-form queuing networks. This introduces problems when we model the system using MVA techniques. However, these problems are well understood and can easily be solved by appropriate modifications of the residence time equation of the exact MVA algorithm.

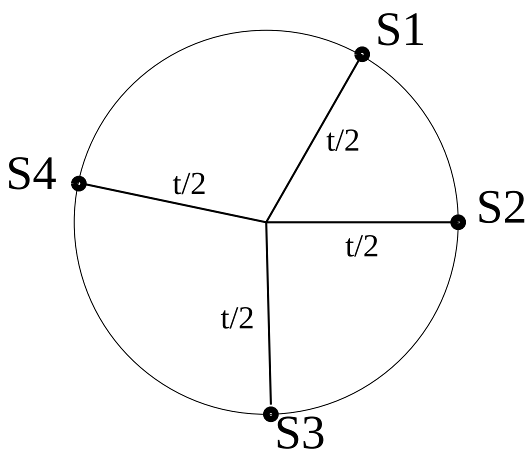


Figure 3.3: A star configuration

each station (other than the sender) is equally likely to be the recipient of an already transmitted request. In other cases we use a mixture of the above schemes. For instance, when a collection of stations are accessing the same server, each of these stations sends its requests to the server, whereas the server replies to each of them with the same probability. Finally, if we deal with an open system, the requests depart from the system as soon as they get across the network successfully.

The above discussion reveals the need for definition of several parameters at the workstation level. First of all we must specify the number of workstations, m, and the amount of processing, D_i, at each of them (i.e. $i = 1, 2, \ldots, m$). If we consider an open system with Poisson arrivals then we must provide the intensity, λ, of the input traffic. Otherwise (i.e., for a closed system) we ought to specify the number of requests, n, that reside in the system as well as the routing policy adopted.

The second important component of the system is the network part. In this part, delays due to interaction and information exchange between the modules of the workstation are considered infinitesimal. If this is not the case (i.e. Physical and Data Link Layer delays and other overheads cannot be ignored) we can incorporate those delays to the processing time, so as to deal only with delays due to network contention at this level.

One of the primary concerns in the network design is its topology which plays an important role to the system performance. In the previous section, we noted a trend towards balanced star networks although this is in contrast with the letter of the Ethernet specifications. This particular topology, where each station is equidistant from a "hub" (where no stations reside) is very convenient for modeling purposes and assures run-time efficiency in our simulations. Since compatibility with existing analytical models and efficiency of execution are two important factors that affected our design, we decided to deal with an equidistant star network. A signal from each station propagates to the passive coupler

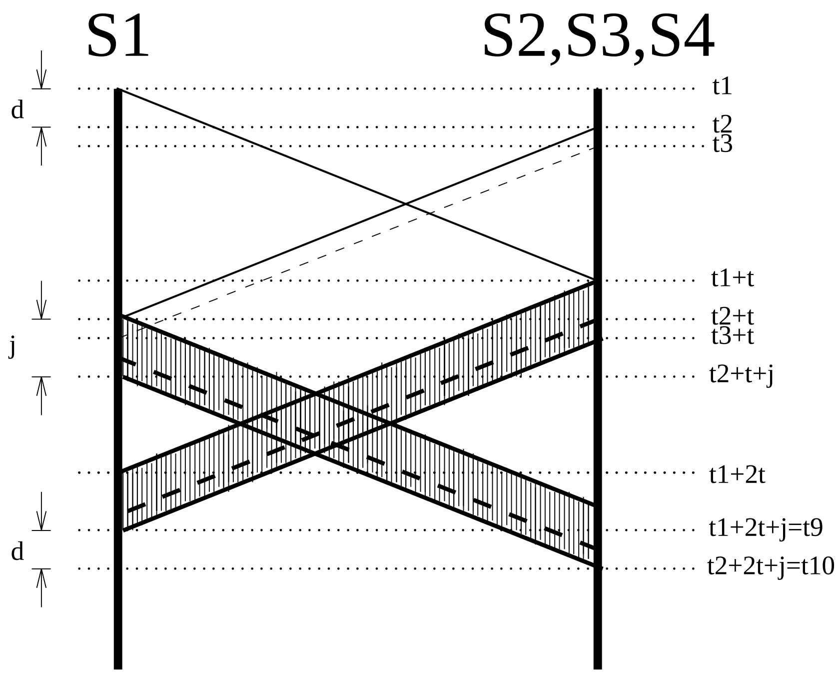


Figure 3.4: Station timing in case of collision

(hub) in time equal to $\tau/2$. Thus, propagation time between any pair of stations of in the network is τ (Figure 3.3).

The frames can be of constant length P, or of bimodally distributed lengths. In the latter case we need to define the "short" length P_s, the "long" length P_l, and the fraction of short packets f_{bd} ("breakdown" ratio). Ethernet specifications impose a hardware spacing between successful packet transmissions as well as between reception of the end of the last jam tone and the next attempt in case of a collision. This gap can be conveniently implemented by incorporating it into the packet size (as an extra 12 bytes) for the cases of successful transmissions, and the jam tone duration j, for the cases of collision fragments, thereby optimizing the number of events processed in the simulator.

Now we will describe the timing of stations in case of a collision by means of Figure 3.4. We consider four stations S_1, S_2, S_3 and S_4, out of which the first three cause a multiple collision. At time t_1 station S_1 starts transmitting a packet on the idle channel. If another station transmits within the "vulnerable" period τ after this first attempt, a collision will occur. In fact, stations S_2 and S_3 make such attempts at times t_2 and t_3, respectively, which come before the end of the "vulnerable" period (i.e. $t_1 + \tau$). At that time, S_2 and S_3 will sense the signal from station S_1, and jam the channel for j seconds. Station S_4 has not detected the collision yet, but now it senses the channel busy and therefore defer any future attempts until the channel turns idle. $\delta \equiv t_2 - t_1$, seconds later both S_1 and S_4 will receive the scrambled signal due to the interference of the station S_2. Station S_1 will start broadcasting a jam tone, whereas S_4 (which now detects the collision) will further defer transmissions. When the signal of S_3 reaches all stations at time $t_3 + \tau$, the multiplicity of the collision increases, but no further jam signals will be generated.

Station S_1 will sense the channel idle again at time $t_9 = t_1 + 2\tau + j$ when it stops receiving the jam signal from the stations S_2 and S_3, and will start backing off at that time instant. The rest of the stations will sense the channel idle and the colliding ones will start counting their backoff intervals at time $t_{10} = t_2 + 2\tau + j$ when the end of the jam tone of S_1 reaches them.

We will call the scenario described in Figure 3.4, a "collision subcycle". Following such a subcycle, the originator station can make its next attempt after time t_9, whereas all other (colliding and non-colliding) ones can do so after time t_{10}. If the first attempt on the idle channel is made by a station other than the originator, a new "cycle" will start. As a result we may have a successful transmission (i.e., no interference from another station) or another collision subcycle where the new originator station is different from the previous one. On the other hand, if the first attempt on the idle channel is made by the same originator between time t_9 and t_{10}, then it is very likely to collide with one of the remaining stations who will start to transmit after time t_{10}.[11] In that case the value of the difference δ, will change in order to reflect the new "timing" of the originator relative to the remaining stations. If we extend the diagram in Figure 3.4, we can derive a formula for the correct update of the value of δ in case of repeated collision subcycles with the same originator. We term such a series of subcycles a *"contention cycle"*.

From the above discussion, one can conclude that in order to derive the correct timing of the network in case of a collision, we need to know the station who first made an attempt on an idle channel, as well as the station who made the next closest in time attempt within the vulnerable period. The first station to transmit will be called the *originator*; it has a special role in the implementation in the sense that it will control the timing of the entire network subsystem. The second station to transmit also plays a role in the timing, by determining the difference, δ, between the time that the originator and the rest of the stations sense the channel idle again after a collision. We must also keep in track the collision fragments, so as to update the value of δ appropriately.

In case that no collision occurs, the signal from the originator will reach all other stations after time τ. After that point the rest of the stations will sense the channel busy and defer packet transmissions until they receive the end of the originator's packet (i.e. at time $t_{12} = t_1 + \tau + \frac{P}{C}$). However (see Figure 3.5) the originator can send its next packet after the successful transmission of its current packet (i.e. at time $t_{11} = t_1 + \tau$). Of course, in all cases the 9.6 μsec interframe gap must be added to the times that the stations sense

[11] This is very likely to happen due to the persistent nature of the protocol. Here we examine the case of a repeated collision. In case of no collision, we will have a successful packet delivery initiated by the originator station.

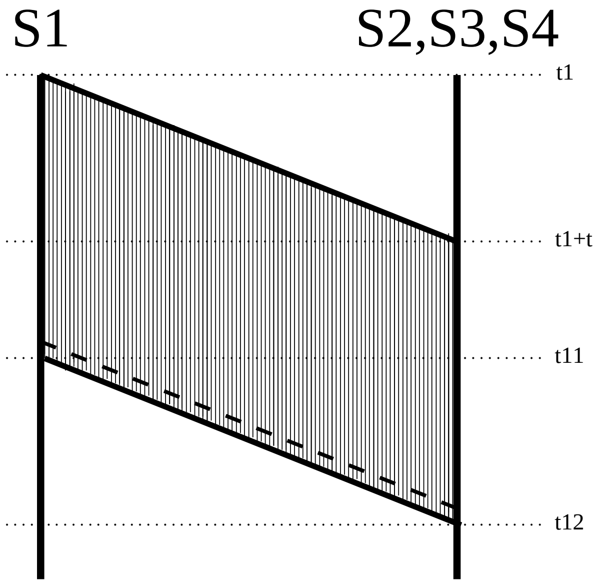


Figure 3.5: Station timing in case of successful transmission

the channel go idle, to determine the earliest times at which the next transmissions can be made (times t_9, t_{10}, t_{11} and t_{12} in the Figures 3.4 and 3.5). As discussed above, we account for the 9.6 μsec gap, by increasing the lengths of the packet transmission time and the jam time, as shown by the dashed lined in the above two figures.

The parameters that we need to define at the network level are the ones that are related to the packet sizes (i.e., either P or the triple P_s, P_l and f_{bd}), the network length (τ), the collision handling (j) and the interpacket spacing (if other than the 9.6 μsec). The slot time for the backoff algorithm is based on the upper bound on the acquisition time of the network, and it is defined to be the sum of twice the maximum end-to-end propagation time (τ_{max}), plus the maximum length of the jam time (j_{max}). Ethernet specification restricts the above two parameters to 23.2 μsec and 4.8 μsec respectively, thus yielding a slot time S of 51.2 μsec.

3.2 Design Analysis

After setting the program specifications, we can proceed to the next stage which is the implementation of the simulation program. A top-down analysis of the program structure and an intermediate level documentation of the event processes are presented here. Some implementation details and problems encountered, as well as some technical considerations, are independently analyzed.

The implementation has been done in the **C** programming language, which provides flexibility in handling and instrumenting various system modifications and collecting a wide variety of statistics, at high execution speed. The simulation model is based on an event-driven structure [54], as this can be accounted by the fact that events (i.e. state transitions of the system) occur at irregular intervals and cannot be predicted in advance.

3.2.1 State Transition Diagram and Event Types

As we already described, the system is composed of the *workstations* and the *network* part. The workstations are functioning like FIFO queues and therefore their operation can be easily simulated by using only two events: **WS_ARRIVAL** and **WS_COMPLETION**. We can define the workstation subsystem *state*, to be its *population* (i.e., the total number of requests at the workstation queues). Arrival and departure of a request are the only reasons that cause the subsystem state to change, therefore the above two events suffice to represent the system behavior. The **C** event routines implementing the above actions are `ws_arrive()` and `ws_depart()` respectively.

The network part of the system is more complex, and requires several additional events (besides request arrivals and departures) to represent the state changes. Here it is more natural to associate the state of the subsystem with the *channel* state. Certain assumptions that we have made help to reduce the number of events, yielding better efficiency. The most important assumption is that stations are equidistant, which means that we can aggregate the workstations into two categories, namely the "originator" station versus all the rest. Evidently all stations can be originators depending on which one first captures the idle channel. This distinguished station has "control" of the channel, in the sense that it determines the time that the various events will take place.

The channel states that we have defined, although they are related to what is happening on the ether, do not directly correspond to the real channel states. The reason is that in the real system, the channel state is different for each station due to its geographical position. Since our model is a "centralized" one, channel state reflects the state of the channel as sensed by the originator. In some cases, however, the state of the channel cannot be described (because there is garbage on the channel). In such cases we use a special state (called **JAM**) that helps to arrange the correct timing of the events and schedule the re-transmissions. Overall, we have defined five channel states: **IDLE, TRIAL, XMISSION, COLLISION** and **JAM**. Figures 3.7 and 3.6 illustrate the correspondence of these states to the channel timing in the cases of success and collision.

The channel is IDLE whenever the originator senses it to be idle. Once the originator starts transmitting, the other stations being τ time units away can continue to schedule attempts until an additional time τ has elapsed. In case a collision occurs (because some other station began transmitting δ time units after the originator), the other stations will lag behind the originating station in sensing the channel idle for time δ (Figure 3.4). Thereafter, when the first station begins a transmission (possibly changing the originator),

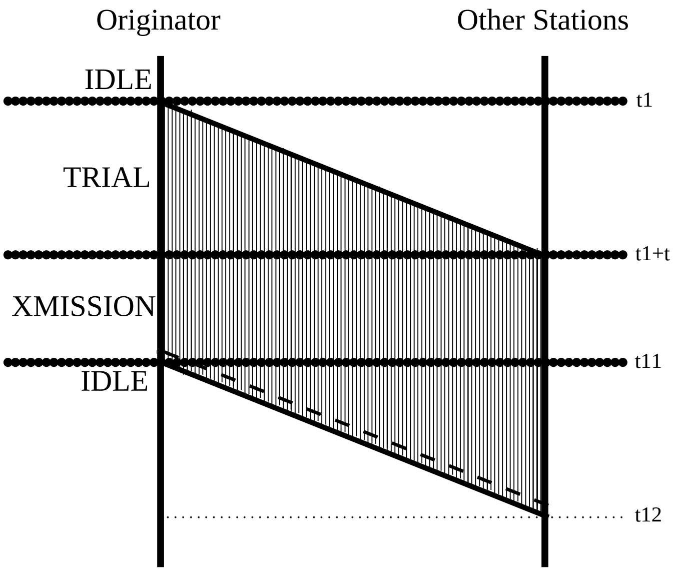


Figure 3.6: Operation of system "states" in successful transmission

the channel state becomes TRIAL and this signifies the beginning of a vulnerable period. If no other station makes an attempt for time τ after the beginning of the TRIAL state, the channel enters the state XMISSION, successfully delivering the rest of the frame, and turns IDLE upon completion of the transfer. Attempts that are supposed to be made by other stations while the channel state is XMISSION are deferred to the instant that they sense the channel turn idle again (which will happen τ time after the moment that the originator switches the channel state to IDLE).

However, if one or more stations other than the originator attempt to send data on the channel during the TRIAL state, then the state changes to COLLISION at the time that the first colliding station makes its attempt. Further attempts (within the vulnerable period) will not change the channel state. At the end of the vulnerable period the originator will test whether the channel state has turned from TRIAL to COLLISION. If such a change was made, the channel state is set to JAM. The originator will turn the channel IDLE at time t_9 and start its backoff at this time. The rest of the colliding stations will start counting their backoff intervals at time t_{10}. Non-colliding stations will defer attempts until time t_{10}. In case that we have another collision subcycle with the same originator (i.e., the contention cycle continues) the new difference $\acute{\delta} \equiv t_{10}^{'} - t_9^{'}$ will be defined by means of the current value of δ and the new difference between the time of the attempt of the originator and the time of the attempt of the first colliding station (i.e. $t_2^{'} - t_1^{'}$). It is important to note that the JAM state includes more than the time that the originator jams the channel. It indicates that the channel is unavailable for transmissions due to propagation of scrambled signals and jam tones, followed by the 9.6 μsec interpacket gap.

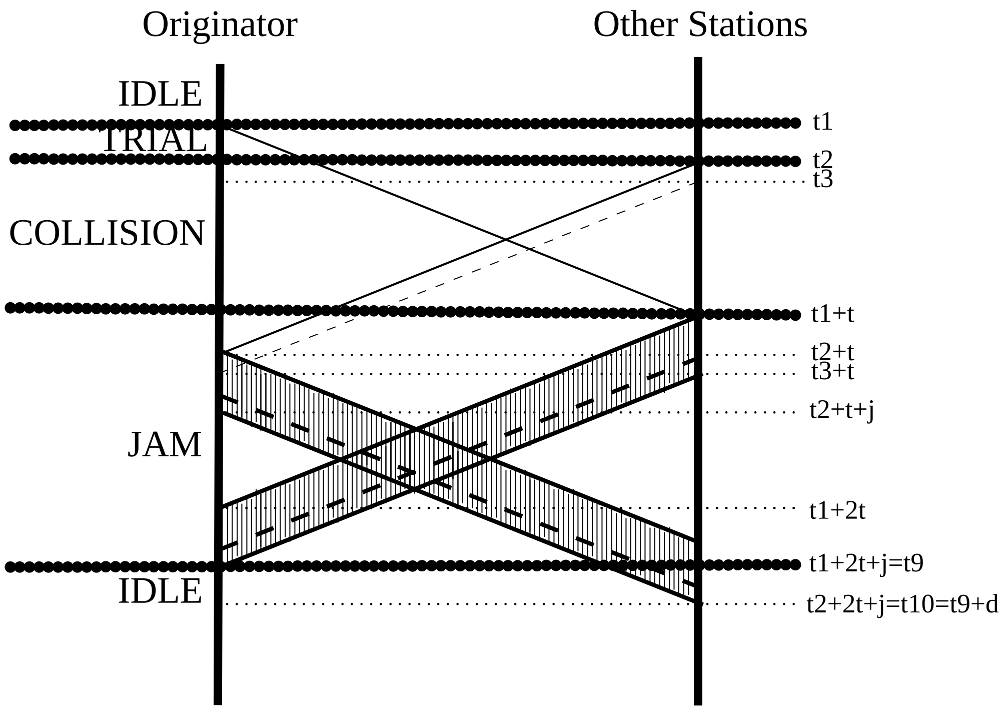


Figure 3.7: Operation of system "states" in case of collision

Figure 3.8 sketches the state transition diagram including the events that cause those transitions, and the name of the respective **C** function that is executed in case of each event. Besides the four events shown in this figure, there is an additional one, which is called **ARRIVAL**, that corresponds to the arrival of a frame at the Ethernet subsystem. This event invokes the **C** function `arrive()`. The **ATTEMPT** event can set the channel state to either TRIAL or COLLISION according to the above discussion. **TRANSMISSION** is the event of checking the channel state τ time after the originator's attempt. **COMPLETION** and **JAM_ OFF** return the channel to the IDLE state after a successful or unsuccessful attempt respectively.

Finally we have considered two more pseudo-events: **ALL_DONE** and **PROGRESS_ REPORT**. The first one is encountered only once at the end of the simulation and the action executed results in the calculation of the final statistics (which is the simulator output) and program termination. The second one is periodically executed, providing information about the state of the system at certain predefined time intervals. It helps checking whether the system has obtained steady state and is a source of intermediate results.

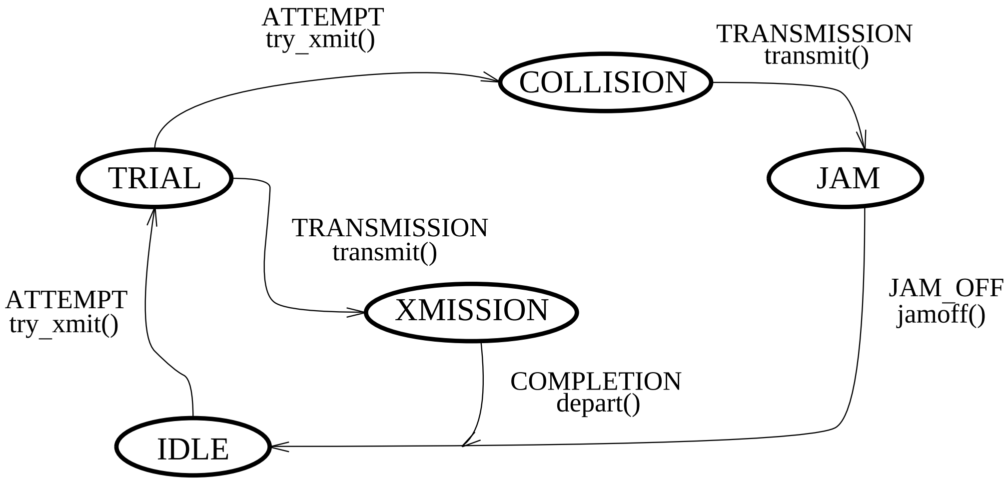


Figure 3.8: Channel state transition diagram

3.2.2 Pseudocode for Event Routines

In this subsection we will go down one level of refinement, analyzing the actions that are taken whenever an event occurs. Because the two events associated with the workstations operation (WS_ARRIVE and WS_DEPART) are similar to the Ethernet ARRIVE and TRANSMISSION events, we will not describe their respective functions separately. We will just point out the differences between the routines `ws_arrive()` and `arrive()` as well as between `ws_depart()` and `depart()`. In the former case (arrivals), the only difference is that in the case of an open system, upon a request arrival, the `ws_arrive()` routine must determine the next arrival according to a Poisson distribution. In the latter case (departures) `ws_depart()` merely sends a "processed" frame to the Ethernet subsystem whereas the `depart()` routine has to implement the routing policy.

In the following description we use the term *contention cycle* which we defined previously as a series of consecutive collision subcycles where the originator station is the same. A contention cycle begins when a collision occurs and ends when either the successful transmission of a frame by some station (not necessarily the originator station) is completed, or when the originator station changes (i.e., during the contention interval some station other than the last originator captures the channel and makes the first attempt on the idle channel). In case of a successful packet transmission, the difference between the time that the originator station and the remaining stations sense the channel idle is τ, whereas after a collision subcycle this difference is equal to δ. The collision fragment entity, helps to distinguish between the above two cases and adjust the value of the difference δ appropriately in case of repeated collisions. The pseudocode for the event routines follows:

```
arrive():

if station queue is not empty then
    frame is queued up;
else
    transmission attempt is scheduled immediately;


try_xmit():

case channel state of
  IDLE      : if not in a contention cycle then /* we have had a success */
                  if the attempt is made by a station other
                          than the last originator and
                      channel entered IDLE less than time τ ago then
                      /* idle channel has not reached the other stations */
                      defer attempt until channel has been
                      IDLE for τ time units;
                  else
                      send frame; set channel state to TRIAL;
              else /* we have had a collision subcycle */
                  if the attempt is made by a station other
                          than the last originator then
                      if the channel entered IDLE less than time δ ago then
                          defer transmission until channel has been
                          IDLE for δ time units;
                      else
                          send frame;
                          channel state is set to TRIAL;
                          this station becomes originator;
                          contention cycle is over (originator changed);
                  else /* Another subcycle with the same originator begins */
                      send frame;
                      originator does not change;
                      channel state is set to TRIAL;
  TRIAL     : if not in a contention cycle then
              /* The last contention cycle must have ended because either a
              successful transmission has been completed or another station
              originated a transmission after a collision subcycle */
                  a collision occurs and a contention cycle begins;
                  channel state is set to COLLISION;
```

```
                    station calculates its backoff interval;
                    set value of δ appropriately;
                else
                /* we have had an attempt with the same originator. Stations
                that were either colliding or deferring know exactly when
                they will make their next attempts. However, there might be
                other stations that could try to send packets before time
                δ has elapsed after the channel became IDLE.
                Note that in this case the channel can turn from IDLE to TRIAL
                instantaneously. */
                    if the attempt is made by one of the other stations and
                        channel has been IDLE or TRIAL for less than δ then
                        defer the attempt until time δ has elapsed after
                        the channel turned IDLE or TRIAL;
                    else
                        a collision occurs;
                        channel state is set to COLLISION;
                        station calculates its backoff interval;
                        value of δ is updated;
    XMISSION  : defer attempts until channel becomes idle;
    JAM       : defer attempts until channel becomes idle;
    COLLISION : attempting station causes a multiple collision
                and calculates its backoff interval;
```

`transmit()`:

```
case channel state of
    TRIAL     : no collision occurred; channel acquired by originator;
                channel state is set to XMISSION;
    COLLISION : channel state is set to JAM;
                originator calculates its backoff interval;
```

`depart()`:

frame has been successfully transmitted;
wake up all deferring stations after time τ and send the next frame
in the originator's waiting queue;
channel state is set to IDLE;
Contention cycle is over (if we were in one);

`jamoff()`:

wake up all deferring stations after time δ and set channel state to IDLE;

3.2.3 Further Implementation Details

Now we will provide some additional hints regarding the correct implementation of the system. In several cases we need to generate random numbers in the interval $[0, 1)$. For this reason we use the custom **C**-library function `erand48()`, which is a generator of random numbers in the above region and guarantees excellent randomization [55]. For every different stream of random numbers we use the above function feeding it with a different seed for each stream. The 48-bit seeds are defined by means of permutations of numbers obtained by the system clock.

The above uniform distribution can be easily converted to other ones (e.g. Poisson) by techniques described in almost all simulation books. The time resolution that we need for the purposes of this simulation is in the order of microseconds, over interval lengths in the order of hundreds of seconds. This can be sufficiently implemented by the **C** type *double* (which provides precision of 15 decimal digits). Normally, integers wherever they are used, cannot cause overflow (note that $maxint = 2^{31} - 1$).

The initialization of a closed system is done randomly, by using a temporary external Poisson arrival process with rate λ equal to $0.8 \cdot \frac{C}{P}$, in order to obtain the first n "arrivals". Then we turn off the external arrivals and let the system run, based on its parameters.

The statistics obtained refer to a wide spectrum of measurements. Besides the average response time and the throughput, we measure the distribution of the channel state, the collision rates, the queue lengths and the second moments of the primary performance measures. In certain cases we have measured the fairness of the system through second moments of statistics obtained by each individual station.

Dropped packets are re-inserted into the Ethernet subsystem and do not require processing at the workstations again. For statistical purposes, their arrival time at the Ethernet remains unchanged. In all cases, they make substantial contributions to the average response time and the variance of it. The program can be easily modified to evaluate any other type of measurements that may be required, revealing the flexibility of the implementation based on the **C** language.

A final remark pertains to the case that multiple events are scheduled at the same time. This is possible due to the balanced star topological layout of the network. In most cases this is very convenient because it eliminates the need for having more events (e.g. JAM_OFF events for each colliding station). However in a very few cases this may cause problems and remedial action must be taken. For instance if we have several ATTEMPT events and a TRANSMISSION event scheduled to occur at the same time, the result will be different in the case where the `transmit()` routine is executed first, versus the case where one of the `try_xmit()` routines precedes execution. In the former case, a collision will be avoided whereas timing considerations in the real system make the collision inevitable. This can be overcome by giving higher priority to the ATTEMPT events (by proper manipulation of the event-list). Nevertheless, simultaneous events in most cases do not alter the simulation result.

Chapter 4

Experiments on Ethernet-based systems

Using the simulation model that we developed, we have conducted a study of the Ethernet behavior in the context of various distributed computing environments. In this chapter we present the results from this series of tests. We have put great emphasis on the issue of network congestion, which can be justified by the increasing speed of today's workstations and communication protocols. Thus, the main assumption in the following experiments is high network demand.

Such traffic conditions can be generated by assuming closed systems with a fixed number of circulating customers and little processing between sucessive network requests. The simplest way to model such systems is to construct separable queuing networks, which have a straightforward and accurate analytical representation. We have modeled the majority of the systems under study via MVA techniques, trying to determine the accuracy of the widely used Ethernet analytic model that we described in section 2.1.1. The Ethernet representation in those models is based on the concept of Hierarchical Modeling, and it can be done by incorporating the entire network subsystem as a FESC that mimics the operation of the entire Ethernet subsystem. Since the function of the remainder of the system is well repesented by an appropriate MVA model, any further discrepancies in the performance measures should be attributed to the Ethernet behavior.

After validating our simulation model against published measurement data, we examine the Ethernet behavior in the typical open system context, and we continue by carrying out various trials in closed system environments and client/server architectures.

4.1 Validation of the Simulation Model

The first set of experiments with our simulation model were aimed at validation. This could be done by comparison of the simulation output with either the results derived by an accurate analytical study of the system or published measurement data from real systems. Since the accuracy of existing models has never been proven, we decided to follow the latter approach and verify the model against measurement results available in the literature. This section is dedicated to the description of the methodology and the analysis of the results. For comparison, it also presents some analytical results that have been obtained using the MVA model from section 2.1.1. Furthermore, it shows the results from an extension of the validation experiment to systems composed of larger numbers of workstations. These extrapolated results, display the limiting behavior of Ethernet under large numbers of stations.

4.1.1 Experiment Description

Amongst the available Ethernet measurement studies, the paper by Boggs et al. [29] appears to be the most suitable for validating our simulator. This is because it provides a detailed description of the network configuration and its characteristics, reducing the number of system parameters that we need to guess or infer in order to represent their particular system with our simulator.

As we described in section 2.2, the system has a bus topology connecting up to 24 high-speed workstations via evenly spaced multiport repeaters. Since the Ethernet specifications restrict the maximum Ethernet segment to be 500 meters long, and the length of the cable was given as approximately 1000 m, another repeater would be needed to connect two 500 m Ethernet segments.

Because our model corresponds to a star configuration, we need to be able to define the "distance" between every two stations (parameter τ) so as to represent the average distance between two stations on a linear bus network architecture. Molle et al. [53] present novel space-time models of asynchronous CSMA systems and provide the right methodology towards the solution of this problem. The paper introduces the notion of *average distance* between two traffic sources drawn at random. The idea can be applied here in order to determine the diameter of a star network, so that it has the same average separation on a bus of length τ. Let $\overline{\tau}$ be the average distance between two uniform points on a bus of length τ. Suppose one of the points is at a distance y from the beginning of the bus. Then the other will be chosen toward the beginning with probability y/τ (in which case the average separation is $y/2$), or else farther from the beginning with probability $(\tau - y)/\tau$ (in which case the average separation is $(\tau - y)/2$). Thus:

$$\overline{\tau} = \int_0^{\tau} \frac{dy}{\tau} \cdot \left(\frac{y}{2} \cdot \frac{y}{\tau} + \frac{(\tau - y)}{2} \cdot \frac{(\tau - y)}{y}\right) = \frac{\tau}{3} \tag{4.1}$$

The one-way propagation time in the bus network under study is approximately equal to 6 μsec. Therefore, the above calculations dictate that a star approximation to this bus network should use a one-way propagation time of $\frac{6}{3} = 2\mu$sec. Some additional delays inserted by the inter-segment repeaters and the DELNI repeaters should also be taken into account. Thus we set the parameter τ to 2.2 μsec in our simulated network. The jam time j is in the range defined by the Ethernet specifications, which is between 3.2 and 4.8 microseconds.

According to the description of the measurement study, there is an interrupt device that services the packet transmission and incorporates a delay in the order of 100 μsec per packet. This can be simulated as processing time at the workstations. Therefore we can set the parameter D to this value. The number of workstations, m, varies between 1 and 24.

Since receiver performance is not an issue in this test, each workstation possesses a single frame and addresses it to itself. The major overhead fields in the frame are the *preamble* and the *Cyclic Redundancy Checksum.* We have modeled those fields by adding an extra 12 bytes of information in the data packets. The system is initialized so that every station holds a single frame (i.e. $n = m$) and can be regarded as a closed system where frames can be of constant or bimodal size.

A disadvantage of the above measurement study that we should point out is that it ignores the impact of queuing in the system response time. What Boggs et al. measured as "Average Transmission Delay" is actually just the service time (i.e. the time between the arrival of the packet at the head of the queue until its successful transmission, which is the delay due to collision resolution time and packet transmission time) for the packet at the head of the transmit queue. This definition neglects the important contribution of queuing at the hosts, which can substantially increase the waiting time in a heavily loaded system. Our simulation model is capable of measuring this effect and we have indeed attempted such a study in the context of client/server configurations which we will illustrate later.

4.1.2 Results

Figure 4.1 shows the simulation results for five different tests on the experimental network, each corresponding to a different packet size. The curves have been labelled by the packet sizes. The results from the measurement study of Boggs et al. have been reprinted in the Appendix on page 108 for the purposes of quick reference.[1] It is easy to see the qualitative and quantitative similarity between our results and the measurement ones. The comparison proves that throughput and response time derived by the simulation are reasonably close

[1]The term *Utilization* that appears in the graph headers in both figure 4.1 and the Appendix is incorrect since these graphs illustrate the *throughput* of the Ethernet subsystem. For the purpose of comparison, we adopt the terminology used by Boggs et al. [29], in this case only. In all subsequent graphs we use the correct term which is *throughput.*

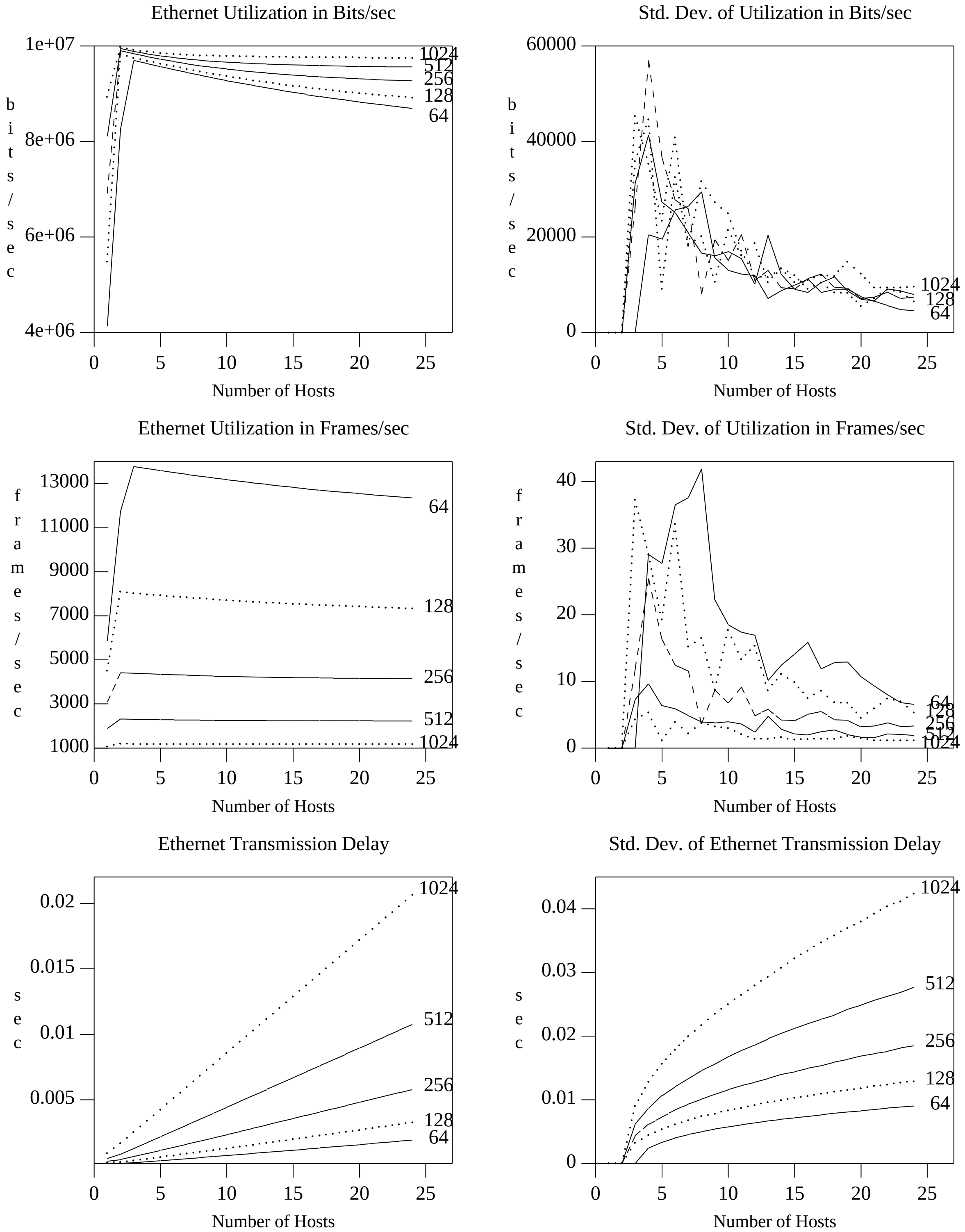


Figure 4.1: Stress-test results from the experimental Ethernet of Boggs et al.

to the real system ones. This was very encouraging, because it displayed the validity of the simulation model.

Moreover, this agreement shows that simulation is an appropriate modeling tool, in spite of network practitioners who question its ability to represent the way that real systems behave. Simulation techniques fundamentally have the expressive power to accurately reflect the behavior of real systems, provided that sufficient information about the system operation is available. Its virtue becomes vital especially in cases where it is infeasible to instrument the real systems for measurement studies, as well as when limitations on the available resources for an experiment (in terms of equipment and time), impose restrictions in the regions of the system parameter space that can actually be measured. Simulation and analytic modeling provide the capability of extending the results beyond these limits with impressive efficiency and correctness.

The only part of the simulation results that did not agree completely with the real system was the standard deviation of both throughput and response time obtained by the per-station statistics. Although those curves were qualitatively akin to the ones obtained by Boggs et al, our results appear to be quantitatively optimistic. This can be explained by the difference in the topology of the two networks. As we noted in chapter 2, Gonsalves and Tobagi pointed out that the distribution of stations affects the aggregate performance and that the individual station performance varies with the station location [18]. Particularly in linear bus networks, performance is in favor of the "central" stations. Evidently this leads to higher variance of the performance measures and provides a plausible explanation to the difference, especially in the standard deviation of the throughput.

In these figures we see that throughput peaks at $m = 2$ workstations except for the case of 64-byte packets, where it reaches its maximum value for $m = 3$. After that point it starts decreasing with the number of workstations due to the increasing collision rates. Small packet sizes incite more collisions (because the arrival rate is increased by the factor by which the packet size is reduced) and therefore the corresponding bit rates are lower. The delays and their standard deviation increase almost linearly with the number of workstations and highly depend on the packet size.

The standard deviation of the throughput demonstrates the fairness of the network and shows that the fairness increases with the number of workstations. Ethernet appears to be unfair for small numbers of hosts due to the intrinsic unfairness of its backoff algorithm. Initially Almes and Lazowska [8] and afterwards other researchers (which we will mention in a subsequent section) pointed out that Ethernet exhibits a degree of last-come-first-served scheduling behavior, increasing the probability that a station with a long backoff defers to other stations which have smaller backoff intervals.

Peaks in the throughput are due to a synchronization effect. In this case two stations are completely synchronized because of the duration of the interrupt service latency which is longer than the packet transmission time and each one of them defers when the other one sends a packet. Hence the two stations never cause collisions. For 64-byte packets this

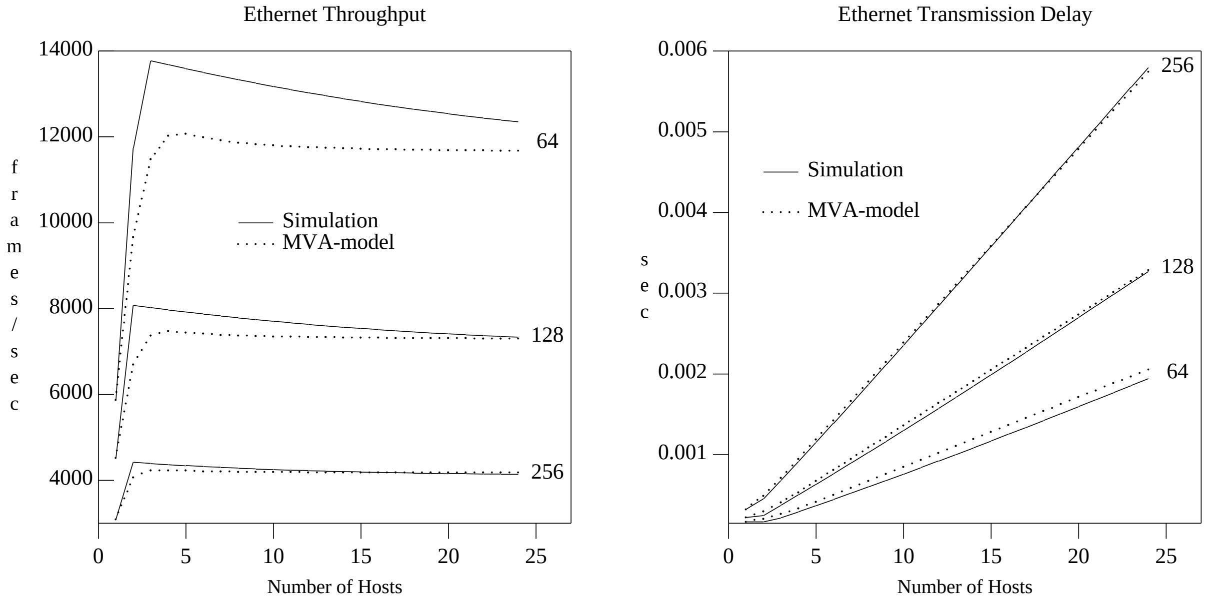


Figure 4.2: Simulation vs. Analysis

synchronization takes place for up to 3 workstations due to the smaller packet transmission time. The above effect will be described in more detail later on in this chapter.

Besides the previous tests, we also tried to evaluate the sensitivity of the result to changes in the network length. The results (which are not presented here graphically for reasons of space conservation) revealed that by changing the length of the network, the shape of the throughput and delay curves in Figure 4.1 is slightly modified. Reduction in the network length enhances the network throughput by a small factor, whereas increasing this length degrades the throughput to a larger extent for systems with more workstations. However, it does not modify the peak points of the throughput curves, which are obtained for systems comprising 2 or 3 workstations (depending on the packet sizes). In other words, the major effect in this case was that the tail of throughput curves in Figure 4.1 was noticeably moved downwards, and this was more obvious for the cases of small packet sizes (i.e., 64 bytes). Thus, the simulation reinforced the analytic results that we described in chapter 2 relevant to the dependence of the Ethernet performance on the length of the network (parameter τ in the analysis) and the size of the transmitted packets. Evidently, shorter networks have shorter "vulnerable" periods (i.e., time interval during which packet transmission attempts by at least two stations results in a collision) which make them less susceptible to conflicts among stations connected via the broadcast medium. Lower collision rates imply shorter backoff delays and therefore enhanced network performance.

Finally we should describe how we constructed a simple analytic model of the above system using QNM techniques. In general, this would require us to use a multiclass model consisting of as many classes as there are workstations, with one service center for each workstation and a FESC to represent the network. Unfortunately this model is not very attractive because of its complexity. But for this system, notice that each class has a population of 1 for its respective workstation, and 0 for all other workstations. The visit count to the network center is 1 for all classes. The service time for the customers of each class at their respective workstation is equal to the interrupt latency (D), independent of the total size of the system. Thus, although it is not a good approach in general to use a single-class model in order to approximate a multiple class one, in this particular case we are fortunate. Using the fact that there is no queuing at any workstation, we can construct a simple single-class model for the above system that is equivalent to the multi-class model. In that model, the "identical" customers have a population equal to the number of workstations (m) and visit counts of 1 at both a delay center, representing the set of workstations, and a queueing center, representing the network.

The results of this analytical model are contrasted with the simulation data in Figure 4.1.2. It can be seen that, as far as the throughput is concerned, the model does not predict the peaks accurately and it does not give good agreement with the simulation for the case of 64-byte packets. The former can be attributed to the synchronization phenomenon that we described previously and which cannot be captured by the analytical model, whereas the latter is an artifact of the poor representation of the existing Ethernet analytic model (via use of the FESC in the above QNM). Indeed Ethernet's backoff algorithm actively participates in that case due the higher collision rates. However in all other cases the model appears to be accurate over this range of parameters, especially in the response time predictions.

4.1.3 Limiting Behavior Under Large Numbers of Stations

We have extended the previous experiment so as to extrapolate the results to larger numbers of workstations. Since the Ethernet specification allows up 1024 workstations, we were interested to see if the results were qualitatively different for large m. Following Boggs et al., we have also considered three packet sizes that exceed the limitation imposed by the Ethernet specification, namely $P = 2048,\ 3072$ and 4000 bytes. The results for the response time, throughput, collision rates and percentage of dropped frames, are shown in Figure 4.3, where curves have been labelled by the size (in bytes) of the transmitted frames.

We have found that the network subsystem collapses at some point when $m \in (200, 500)$. Beyond this point, the collision rates become extremely high and the host stations spend all of their time counting their backoff intervals rather than transmitting useful information. This collapse is surprising, since it is well below the 1024 station limit imposed by the Ethernet specification. We can also see that the "critical" point (i.e., the one beyond which instability is observed), depends on the size of the transmitted packets. We also

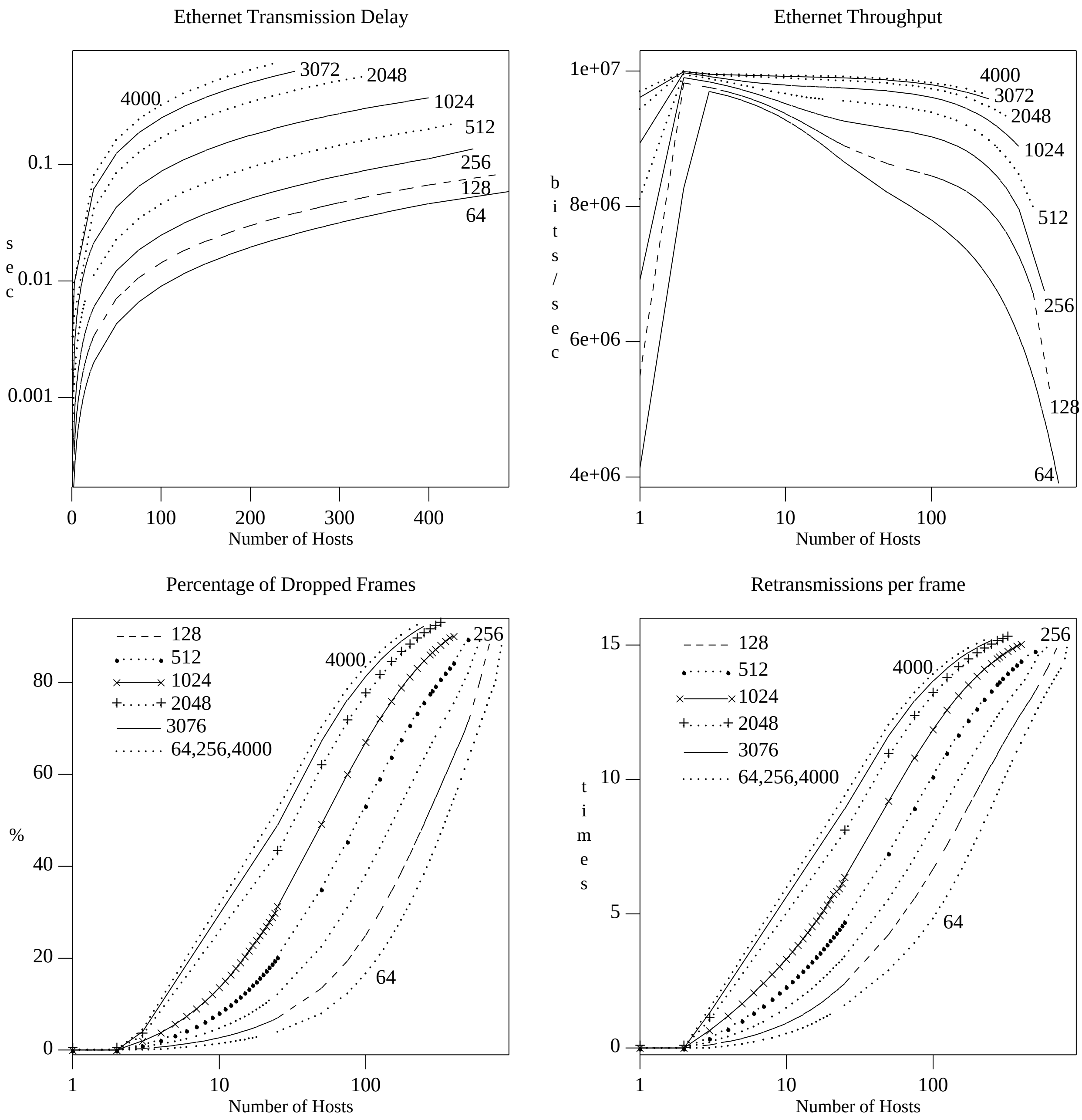


Figure 4.3: Stability of the experimental Ethernet of Boggs et al.

observed that the collision rates as well as the percentage of dropped packets are higher for the cases of larger packet sizes. This is attributed to the fact that when the packet transmission time is longer, it is more likely that more stations will become "ready" during a busy period and therefore more collisions will occur during the subsequent idle period. Particularly, for packet sizes larger than the maximum allowable size (i.e., 1536 bytes), the point after which we observe instability corresponds to only 200 workstations, thus showing the importance of restricting the maximum size of the transmitted packets in the Ethernet specification.

It is worth mentioning that a similar result is reported by Molle and Konstantas [22]. In this paper, the authors assume an open system with short packets and show that this system becomes unstable when the load is high enough. Ethernet is simulated there as a special case of the Virtual Time CSMA protocol by setting the parameter η to a very large value. The operation of that simulated system violated the normal Ethernet protocol in that the stations were reordering their transmit queue after a backoff, which could actually increase the collision rates in the at system. The particular paper pointed out that this behavior was like an Ethernet system with 21 DELNI multiport repeaters each with 15 unbuffered stations, for a total of 315 "virtual stations".

The unstable behavior of Ethernet in the region of a few hundreds of workstations can be explained analytically, if we consider a cycle consisting of the sequence of backoff intervals that a station makes during repeated attempts to send a packet. The maximum cycle length occurs when the station makes 16 attempts, all of maximum backoff, before it drops the packet. Assuming that every attempt is a collision, and the collision time is approximately equal to the slot time, the maximum cycle time is given by:

$$1 + 2^1 + 2^2 + \ldots + 2^9 + 2^{10} + 2^{10} + 2^{10} + 2^{10} + 2^{10} + 2^{10} = 7167 \text{ slot times} \tag{4.2}$$

Therefore, since the actual backoff times are uniform between zero and the maximum, the average cycle length is half of the length of the maximum cycle, or $7167/2 = 3583.5$ slot times. Since the cycle contains 16 transmission attempts, the average time between transmission attempts by the same station is: $3583.5/16 \simeq 225$ slot times. If we have a success by the given workstation, then we drop out some of the big parts that correspond to higher order collisions, thus making the average time between transmissions over the cycle shorter on average. Furthermore, if we have a success by another station then several of these cycles (each one corresponding to a different station) are combined together, and this acts like fewer "steps" in the above cycle, thus reducing again the average time between transmissions.

4.2 Ethernet Behavior in Open Systems

Most of the existing CSMA/CD analytic studies make the assumption of Poisson input traffic. That is, the Ethernet is considered to be a part of an open system where requests arrive according to an exponential inter-arrival time distribution and depart upon service completion. In this section we investigate the behavior of the Ethernet subsystem in such an environment. Our motivation for this is two-fold. First, it is a good starting point for evaluating Almes and Lazowska's Ethernet analytic model[2], since service centers in a separable QNM must obey device homogeneity (sometimes described as "on-line behavior equals off-line behavior").

Second, we wish to study the inter-departure time distribution under Poisson input traffic. Again, this is interesting in the context of separable QNMs due to the required $M \Rightarrow M$ property [56, 35]. In addition, it allows us to re-evaluate a common conclusion in Ethernet traffic measurement studies that Ethernet traffic is not representable as a Poisson process because their empirical traffic distributions are non-exponential. However, the flaw in the argument is that measurement studies record *departure* rather than *arrival* times, and concluding that non-exponential departure times imply non-exponential arrival times is not logical.

4.2.1 Experiment Description and Results

We consider a star network consisting of m identical workstations. The distance between any pair of stations is $\tau = 23.2$ μsec and the jam signal duration is $j = 4.8$ μsec, reflecting the worst case star network vis-a-vis the Ethernet specifications. Packet sizes are bimodally distributed with the "low" peak at $P_s = 132$ bytes and the "high" peak at $P_l = 1096$ bytes. The proportion of short to long packets is $f_{bd} = \frac{1}{3}$. Assuming a $C = 10$ Mbps network, the above leads to an average packet size $P \approx 774.66$ bytes whose transmission time in the order of 0.6 msec. There is also an overhead of 12 bytes of extra information in each frame as well as interframe space of 9.6 μsec.

The input traffic consists of data packets arriving at the workstations with exponential inter-arrival times. In our simulation we define the offered load ρ (attainable utilization) in order to calculate the parameter of the exponential distribution $\lambda = \rho C/P$. In fact this distribution is an aggregate of Poisson processes each with rate $\lambda_i = \lambda/m$, that determine the arrivals at each station $i = 1, 2, \ldots, m$. We vary the parameter ρ in the region $[0.05, 0.95]$ and the number of stations m in the set $\{1, 2, 3, 5, 10, 20, 50, 100\}$. No processing other than network service time is assumed (i.e., $D_i = 0, \ \forall i$).

The results for the throughput, response time and queue lengths are shown in Figure 4.4, where curves are labelled by the number of workstations. The results are also contrasted with the analytic model solution. Furthermore, we provide some additional graphs in Figure 4.5.

[2]Sohraby et al. [16] incorporate much more of the detailed behavior of Ethernet than Almes and Lazowska's model. Unfortunately, their results only give expressions for throughput (or equivalently, the required number of transmissions for a given input rate), whereas Almes and Lazowska's model give response times as well.

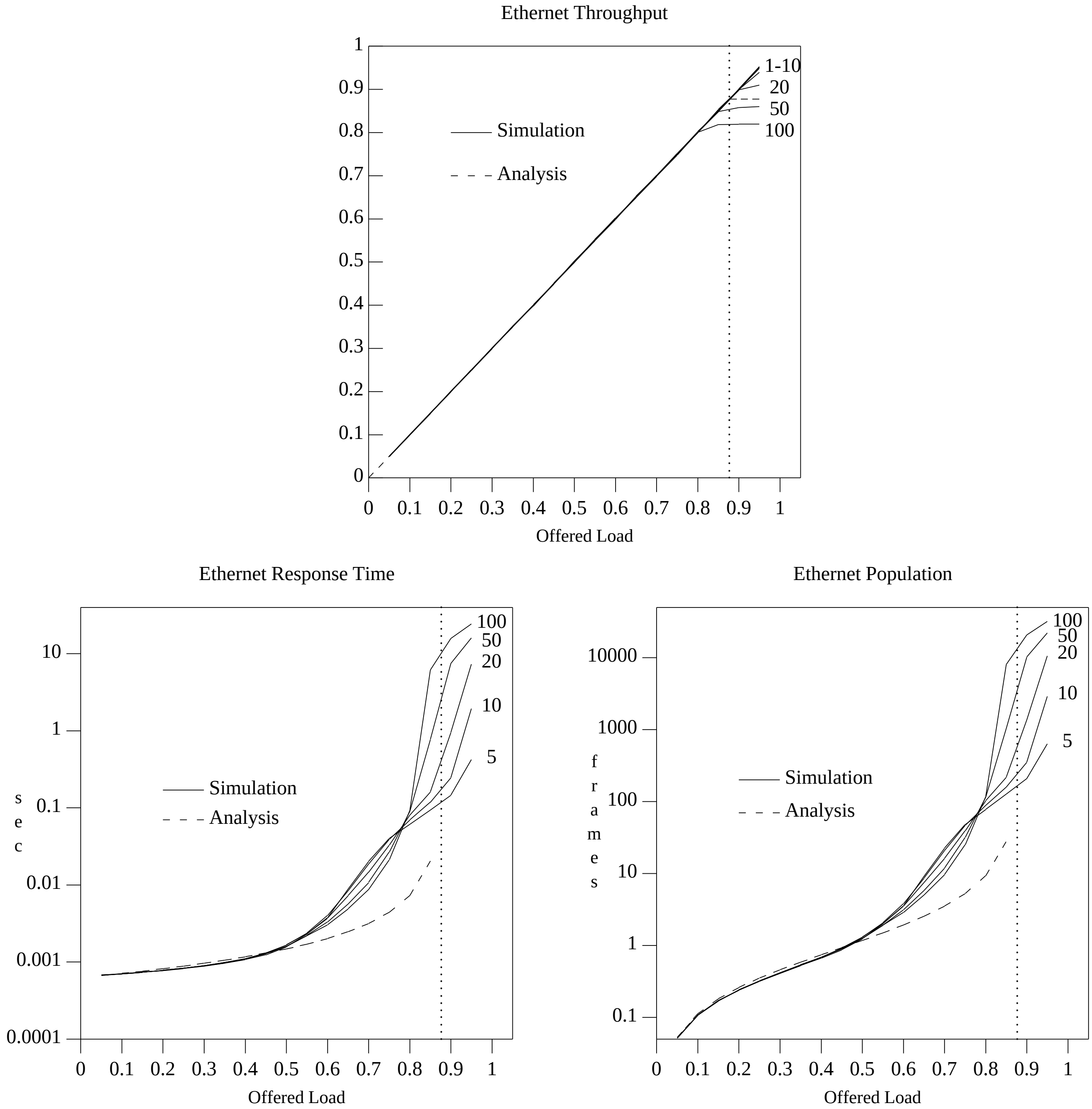


Figure 4.4: Ethernet performance in an open system (a)

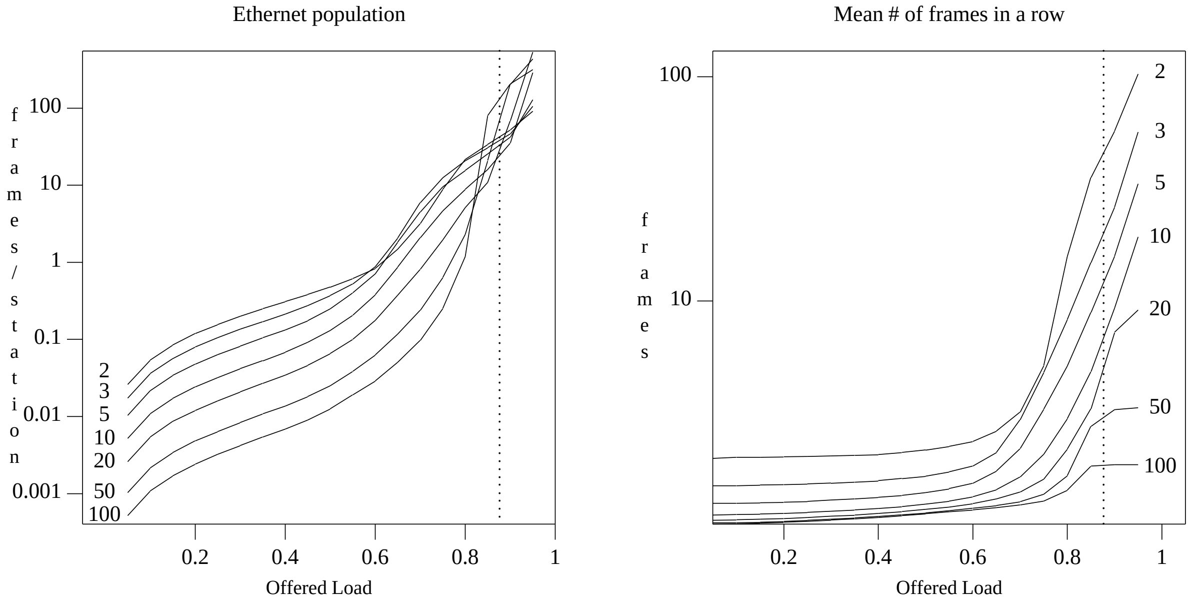


Figure 4.5: Ethernet performance in an open system (b)

We solved the birth-death model that we presented in section 2.1.1 by means of the *Global Balance* technique via use of the *Generalized Birth-Death Equations* [36]. These formulas yield the probability p_q of the system being at each state q in the model. A problem arises due to the infinite number of states and the state dependent service rates in that we cannot find closed form solutions for the p_i's. However the problem can be solved if we assume a finite number of states, say ς, and approximate the tail.

Particularly, we observe that this model has state-independent arrivals at rate λ and state-dependent departures at rate μ_q which is a monotonically decreasing function of q with limiting value μ_∞, which is given by Equation (2.4) with substitution of $W(q)$ by its asymptotic value, $e - 1$. In order to bound the tail we assume that $\mu_\nu = \mu_\varsigma$, $\forall \nu > \varsigma$ for the upper bound, whereas $\mu_\nu = \mu_\infty$, $\forall \nu > \varsigma$ for the lower bound. This technique gives closed-form summations for the tail. We observed that for large ς (i.e. $\varsigma = 200$) the above bounds converge to the same value, which we have used for the analytic solution of the model. The vertical dotted lines in Figure 4.4 signify the analytic model predictions for the network saturation point.

The throughput curves appear to be as expected. All configurations have almost identical behavior in the region $\rho \in (0, 0.75]$ before the network becomes saturated. This is the expected behavior due to the *flow conservation law* which states that the system throughput should be equal to the arrival rate at the system prior to saturation. The behavior of the saturated systems (i.e $\rho \in (0.75, 0.95])$ is qualitatively different with the system throughput degrading monotonically as we distribute the load among more stations. It is obvious that different configurations have different saturation points, as opposed to the analytic model that predicts a unique saturation point. In fact the above analytical model shows that asymptotic behavior appears at $\rho = 0.88$, whereas simulation shows that a 100-station system saturates at $\rho = 0.75$ and a 10-station one at $\rho = 0.9$.

This indicates one of the weakness of the analytic model, which is its inability to distinguish between systems formed by different numbers of workstations. Instead it assumes that there is an infinite backlog of workstations so that all of the customers in the system that wish to use the Ethernet compete simultaneously. However what happens in reality is that only the customers at the head of each station queue contribute to the network contention.

As far as the other two performance measures are concerned, we note that the response time and the queue length curves in the logarithmic scale show some peculiar overcrossings indicating for instance that in the range $\rho \in [0.6, 0.75]$ the 100-workstation architecture slightly outperforms the 10-workstation one. On the other hand increasing the load to the range $\rho \in [0.8, 0.95]$ the situation is reversed and the latter system gives better performance.

Figure 4.5 provides an explanation for this behavior. We can see that in the former region (i.e., $\rho \in [0.6, 0.75]$) the average queue length at each station is approximately less than one frame for systems composed of $m \geq 10$ workstations, and greater than one frame for systems with fewer stations. According to the Ethernet protocol only the frames at the head of the transmit queue of each station are considered for transmission. If those frames experience massive collisions, all other frames in the same queue are delayed too. Therefore, the long backoff intervals of the packets at the head of the transmit queues, can make significant contributions to the overall network response time. However, this is not the case for the "large" systems, where practically no packets are stuck behind the ones at the head of the transmit queues.

On the other hand, in the subsequent range (i.e., $\rho \in [0.8, 0.95]$), "smaller" systems appear to have shorter queues. This observation, in conjunction with the *channel monopolization phenomenon*[3] makes systems with fewer workstations perform better. Figure 4.5 clearly shows that this is the case for the 2-5 workstation systems, where each station after it captures the channel, sends on average as many packets as its average queue length, consecutively.

The conclusion that can be drawn from the comparison with the analytic model is that the model performs very well (in terms of the response time) in the range $\rho \in (0, 0.5]$. However when the utilization is above 50% the analytic results appear to be optimistic with increasing divergence for higher values of the utilization. One reason for that could be the contribution of the dropped packets to the delay, which the analytic model doesn't

[3]This phenomenon is attributed to the binary exponential backoff algorithm and applies to cases of heavy traffic load. Under those circumstances when a station captures the channel, it monopolizes it by sending all packets in its queue before another station takes over. We will elaborate on this, in section 4.3.1.

account for. Moreover, unlike its representation in the analytical model, the real backoff algorithm is not work conserving, in the sense that the channel can remain idle although there are frames wishing to be transmitted at the station queues. This is another artifact of Ethernet's backoff algorithm, which delays all frames that pile up behind the one that is currently being considered for transmission. If the backoff algorithm makes this head frame experience a long backoff, then the channel is essentially "starved".

In all, the observation was that the backoff algorithm behavior which broadly manifests itself especially at high loads *is not* captured by the existing analytic model. The backoff algorithm is responsible for the deterioration of the response time at high loads, and not the access protocol, which the analytic model handles very efficiently.

4.2.2 Inter-Departure Time Distribution

In the previous section we were assuming exponentially distributed inter-arrival times. As we mentioned in Chapter 2, many researchers have tried to characterize the actual inter-arrival time distribution based on measurements on real systems because this is one of the factors in designing networks and specialized network servers. The measurement results have concluded that the above assumption, despite its popularity in analytic and simulation studies, is inappropriate to model real Ethernet traffic due to dependencies in the packet arrivals.

However, those measurement studies (like Gussela [25, 26, 27]) cannot really measure inter-arrival time distributions since all they can see is inter-departure times, after the combined effects of the access protocol, backoff algorithm, etc. Analytic models are based on Poisson arrivals, not departures. Since it is not easy to instrument the network to record inter-arrival times, the next-best way to test the hypothesis of Poisson arrivals is to feed Poisson input to the simulator and compare the inter-departure time distribution from the simulation output to that of the measurement studies to see if the measurements could have come from a Poisson model.

The issue of inter-arrival time distribution is certainly an interesting problem. However the results are applicable only to specific system environments and particular applications. Since we do not want to make our results too application-specific, we decided to look at the output process that corresponds to Poisson input, due to its theoretical importance, namely the $M \Rightarrow M$ condition in product-form queueing networks [35]. The $M \Rightarrow M$ property holds for a queue if the output process that corresponds to a Poisson input process is also Poisson. This reveals another significance of the investigation of the inter-departure-time distribution.

Since we do not assume PS or LCFS preemptive-resume queuing discipline, the $M \Rightarrow M$ property requires exponential service times too. Unfortunately, this is not the case for our system, because we consider bimodal packet sizes and therefore the dominant part of the service times, especially under high loads, is vitally affected by this assumption. Thus its is guaranteed that the inter-departure time distribution will not be exponential. According to their definition, inter-departure times include the packet transmission times which is a non-exponential component. If we neglect this component, the remaining part in an inter-departure time, is the time between the successful transmission of a packet and the beginning of transmission of the next packet, which is called inter-packet time. Therefore it would be interesting to figure out how closely the distribution of the inter-packet times resembles the exponential distribution.

We found that the Ethernet binary exponential backoff collision resolution algorithm had a significant effect in distorting a nice Poisson arrivals process in the sense that exponentially distributed inter-packet times, when used to feed the input to the Ethernet protocol, resulted in non-exponential inter-departure times. To draw this conclusion we followed the method of the *curve-fit-test*, described by Law and Kelton [57], in order to compare the interpacket time distribution (derived from the samples that we obtained from simulation runs) to an exponential distribution with the same mean.

We applied a goodness-of-fit hypothesis test in order to visually compare the data (samples) with the density of an exponential distribution. We divided all sampled inter-departure times into k buckets. The buckets may be of constant size, although that would make the comparison pretty difficult because the shape of the *probability distribution function (pdf)* of the sampled distribution would depend on the value of k. Since this test was a visual test, we had to be as confident as possible for the validity of it. According to a few generally accepted guidelines, it is recommended that the buckets should be chosen so as to equate the probability mass associated with each one, which results in variable-sized buckets. The main reason for choosing the buckets in this fashion is because it makes the test insensitive to the number of buckets k. This choice of the size of buckets means that the plot of both the *pdf* and the *cdf (cumulative distribution function)* of the exponential distribution in a diagram where the X-axis represents the bucket number would be straight lines; a line parallel to the X-axis for the former and the 45° line for the latter. We have used $k = 32$ buckets in our comparison which is a widely acceptable number because it provides sufficient accuracy.

In general, equalizing the probability mass across k buckets, requires finding the "cut-off" values $c(i)$ for each bucket i, such that

$$\mathcal{F}(c(i)) = \frac{i}{k}. \tag{4.3}$$

holds, where $\mathcal{F}$ is the cumulative distribution function. In our case, $\mathcal{F}$ represents an exponential distribution with mean $1/\lambda$, so that $c(i)$ is the solution to:

$$\frac{i}{k} = 1 - e^{-\lambda \cdot c(i)} \tag{4.4}$$

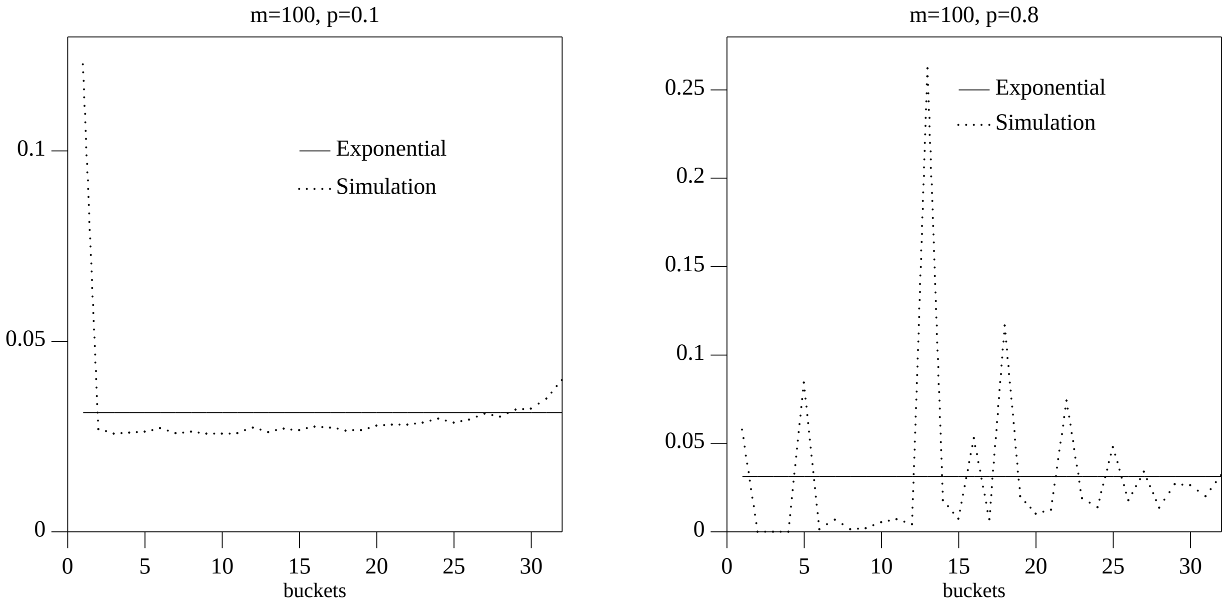


Figure 4.6: Inter-packet time distribution

Solving (4.4) we gain a closed formula that determines the cutoff values, which is:

$$c(i) = \frac{1}{\lambda} \cdot (ln\ k \quad - \quad ln\ (k-i)). \tag{4.5}$$

Now we can arrange the samples into buckets defined by (4.5) to see whether they are equally distributed amongst them. To optimize the partition we can use a binary search scheme that yields the bucket hit by any sample in approximately $log_2 k$ comparisons. If the samples constituted an exponential distribution, the pdf obtained by these buckets would be a horizontal straight line parallel to the X-axis.

We repeated the above test for various values of the parameter $\rho = \lambda/\mu$ comparing the distribution to the exponential with parameter λ. When the Ethernet utilization was 10%, we found that instead of a straight line we got an almost straight line with a noticeable elevation in the range corresponding to the first few buckets. This is attributed to packet deferrals due to the 1-persistence nature of the Ethernet protocol. As the utilization increased the distortion became serious and could be seen all the way along the horizontal "exponential" line. We also measured the distribution for utilization equal to 80% observing an extremely high distortion. For this case, the pdf curve resembles a piece-wise linear interpolation of several points across the horizontal "exponential" line, forming a number of "spikes". We have found that most of these spikes correspond to multiples of the slot time $S = 51.2\mu$sec.

These results can be seen in Figures 4.6 and 4.7. The second figure also illustrates a comparison of the simulation results with measurement results obtained by traces of

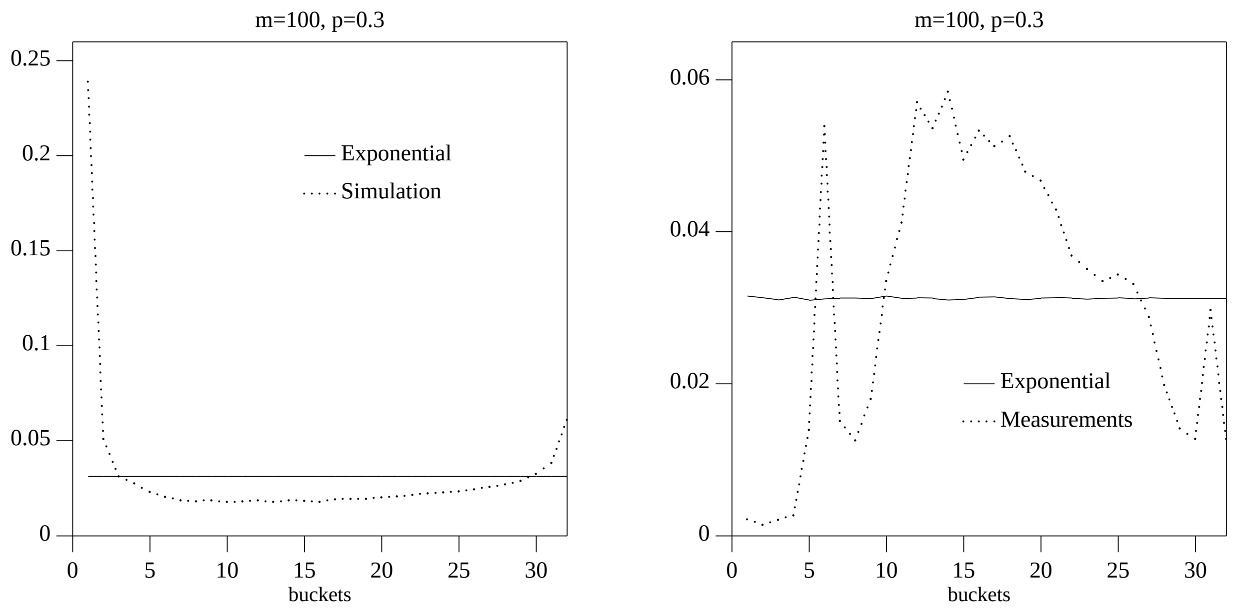


Figure 4.7: Comparison to measured inter-packet time distribution

the diskless workstations network that exists in the Department of Computer Science at the University of Toronto. The number of interconnected stations is approximately 100 and the measured network utilization is 30%. These measurement results were the only immediately accessible ones and although they are not very accurate because of reasons that we explain afterwards, they provide a fair idea of the real Ethernet traffic.

The measurements have been conducted via use of *"EtherTap"*, which is a modified Ultrix Operating System Kernel running on a VaxStation 3200 PC.[4] The utility timestamps the packets as soon as they arrive at the host Ethernet interface. It also extracts the packet lengths and uses this information to calculate the inter-packet time. We have manipulated this data so as to construct the curves shown in Figure 4.7.

We should point out a few problems that were encountered in the measurement experiment that might introduce discrepancies in the result. First of all when the system is highly loaded it uses buffers to hold the packets that the host interface cannot see immediately, and therefore there is a lag between the actual packet arrival and the time that it is "tagged" by the utility. Another disadvantage is that this measurement tool is not installed at the transceiver or at the Ethernet cable. Thus it ignores a significant amount of information in cases of high collision rates. Furthermore, the maximum value of latency

[4]The utility was developed by Michael Galler, a student of Professor Sognian Zhou, for the purposes of characterizing the load across the network of these diskless workstations and perform load balancing. The measurement experiment was carried out by Sam Vuong, a student of Professor Mart L. Molle as an extension to an Operating Systems course project.

in the software utility that issues the timestamps was comparable to the 9.6 μsec inter-packet gap imposed by the Ethernet specifications. This implied that the utility could not clearly "observe" back-to-back packets. For these reasons in our comparison we have considered a lightly-loaded system in order to get a representative view of its operation via this measurement tool.

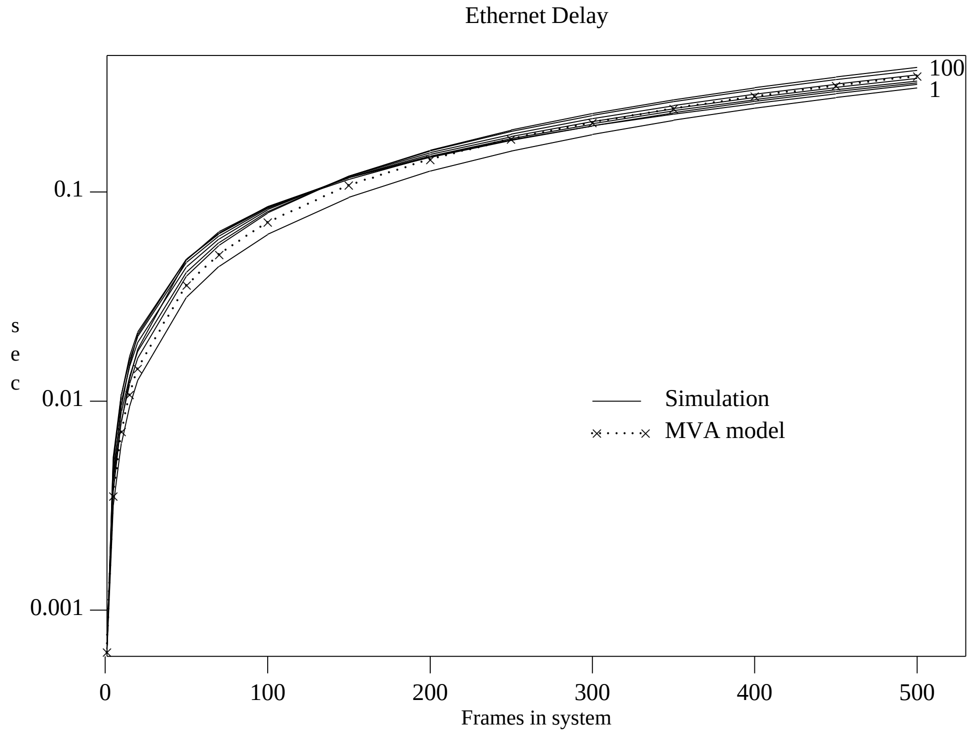
The comparison shows that in both cases (i.e., simulation and measurement) the inter-packet time distribution is non-exponential. The measurements curve looks more like the simulation curve for $\rho = 0.8$.[5] The difference between the simulation and the measurement curves can probably be justified by the "star" topology of the simulated network as opposed to the topology of a real network where the various stations are not equidistant. Also the distribution of packets transferred over the real network is different from the simulation one. Usually real systems carry more packet sizes than the P_s and P_l that we consider in the simulation model. Moreover, the accuracy of the measurement results is questionable especially in the region that corresponds to the first few buckets (where accuracy requires very high-resolution). Overall, this experiment shows that non-exponential distributions that can be seen by measurement studies on real Ethernet environments, might come from a Poisson input traffic model. This shows that we should not reject the assumption of exponential inter-arrival times, which is very popular in analytical and simulation studies, merely because traffic measurements show non-exponential inter-departure times.

4.3 Ethernet Behavior in Closed Systems

As we described in section 3.1.2, our simulation model can easily be modified in order to mimic the operation of the Ethernet LAN in a closed queuing network, formed by m identical workstations and a fixed number, n, of circulating requests (i.e. frames demanding transmission over the network). Each frame is initially lined up for processing at one of the workstations. When the processing is complete the frame is queued for transmission on the Ethernet with the destination chosen randomly from among the other workstations. The amount of processing per visit to a workstation is the same for all workstations and is denoted by D (sec). In cases where a frame fails to be transmitted successfully after 16 attempts it is inserted at the tail of the transmit queue of the same workstation in order to be retransmitted (but it does not receive processing again).

In this section we illustrate the results of experiments that we carried out in such a closed system. The network specifications are the same as in the case of the open system

[5] It would be worth pointing out that the measurement result for Ethernet utilizations in the order of 10%-15%, resembles the simulation curve presented in Figure 4.7 to a larger extent. This is because when the utilization of the network is low the traffic can practically be attributed to packets due to a certain transmission protocol such as NFS. Thus the packet sizes in this case are similar to the ones that we use in our simulation, in the sense that they are not mixed with packets due to other communication protocols.



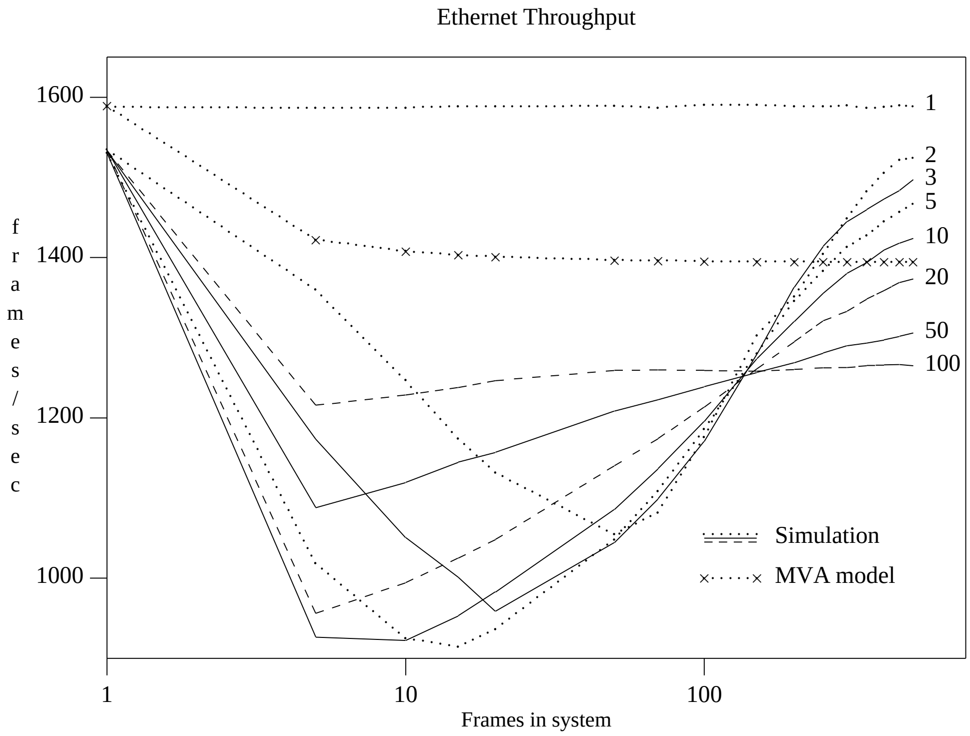


Figure 4.8: Ethernet Performance in a closed system with D = 0 (a)

that we examined previously, in terms of topology, packet sizes and carrying capacity. We have tested systems consisting of $m \in \{1, 2, 3, 5, 10, 20, 50\}$ workstations, for populations $n \in [1, 500]$ and processing times $D \in \{0, 0.001, 0.005, 0.1, 0.2\}$ seconds.

Starting from the description of the results in the case of no processing delay at the workstations (i.e., $D = 0$), we will present some efforts towards analytic modeling of the Ethernet subsystem using MVA techniques, and describe a slight variation of the Ethernet protocol that leads to enhanced performance. Furthermore we will investigate the (more realistic) case where a small amount of processing time at the workstations is involved.

4.3.1 "Stress-testing" the Ethernet

Here we are considering the above closed system where the service time at the workstations is set to zero, and therefore the frames are repeatedly requesting transmission as soon as they are delivered. Although this case seems to be unlikely for real systems, it can actually occur when Ethernet traffic is characterized by bursts of multi-packet trains. This could be a result of bulk data transfers, with lots of storage on the Ethernet board, where big chunks of data are fragmented into multiple packets (each satisfying the Ethernet packet size specifications) that are then sent "back-to-back" over Ethernet.

Figures 4.8 and 4.9 show the plot of the performance measured for this particular system. The horizontal axis in all graphs denotes the population, i.e., number of frames in the system. The curves are labelled by the number of workstations in the system. In terms of delay, we observe that the response time curves have a "hyperbolic" shape in the region $n \in [1, 130]$ whereas in the subsequent range, $n \in (130, 500]$, they show a linear increase. This linear asymptotic behavior can be explained by Little's law. We can see that after this point (i.e. $n > 130$) the network throughput practically stops changing, so response time is a linear function of n.

However, the throughput curves reveal an unexpected behavior of the Ethernet subsystem. We can clearly see that the system throughput starts out at its theoretical maximum (a fully utilized network–which is represented by the dummy case of 1 workstation addressing packets to itself, in these graphs), then makes a rapid decrease in the region $n \in [2, 50]$ and finally starts increasing until it levels off after some saturation point.

The variance of the Ethernet delay increases with the number of frames in a manner denoted by the concave shape of the curves in Figure 4.9, and it is higher for systems comprising more workstations. For $m = 100$ and $n = 500$, the variance is almost half the mean. We also observe that there is a "convex" increase in the percentage of dropped frames as their population increases. This percentage is significant for systems composed of more than 20 workstations, since it is on the order of 5-15 % of the total frames.

The bottom two graphs in Figure 4.9 apply to the study of the *channel monopolization* or *channel capture* phenomenon that has been described in many theoretical studies [13, 58, 59, 60]. This situation is usually applicable to heavy traffic conditions such as

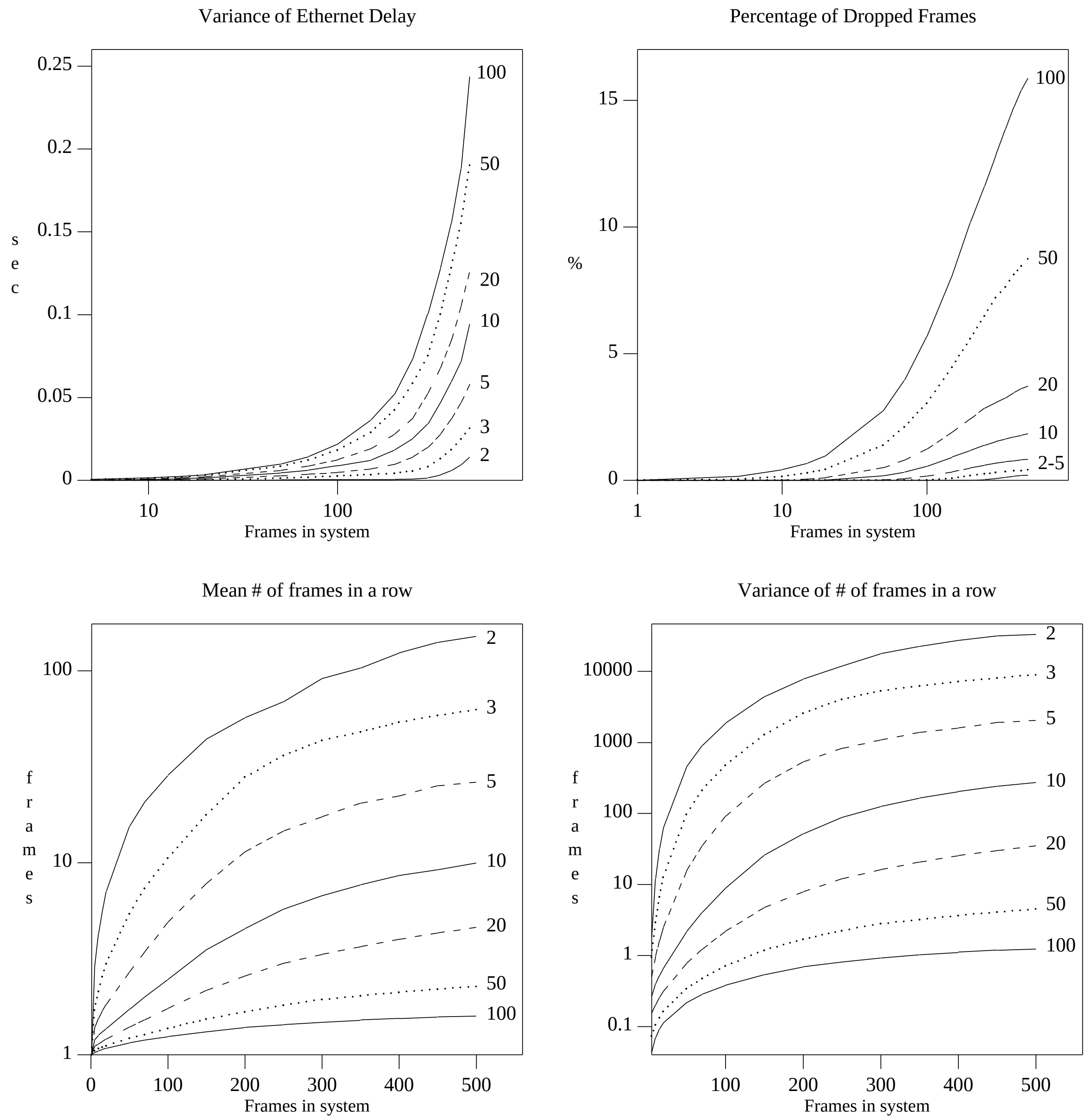


Figure 4.9: Ethernet Performance in a closed system with D = 0 (b)

the one that we have considered in this experiment where the generation of new packets immediately follows the successful transmission of other ones without any further delays at the workstation centers.

Under heavy load circumstances it is very likely that all stations have packets in their transmit queues. Eventually only one station will capture the channel, after winning the "backoff" race, while the packets at the head of the transmit queues at all other stations will be blocked. These stations will schedule their next attempts according to the retransmission algorithm. However the "successful" station has higher channel acquisition probability than the other network stations that are backing off, since its backoff counter is set to zero. In case that the same station makes another successful transfer, it is even more likely to acquire the channel instead of one of the other stations during the contention period. In this way while all other stations are stuck in a long backoff, a station that captures the Ether tends to empty its queue, then another station takes over and so on.

This phenomenon can be attributed to the behavior of the Ethernet's binary exponential backoff algorithm and was widely observed in our simulation test. We have measured sequences of repeated transmissions by the same stations and reported the mean and the variance of the length of those sequences. It can be seen that the *channel capture* phenomenon is in effect and it is more intense for systems comprising a few workstations, whereas it decays for larger configurations. The extremely high variance of the sequence length verifies that individual stations can hold the channel for long periods of time, and thus demonstrating the unfairness of Ethernet when there is a small number of active stations.

In cases when the average number of frames per station was less than one, the lack of processing time led to a similar phenomenon that could be described as packet monopolization. In those systems, we found that a packet is passed from station to station along a routing "chain" until it gets stuck behind a packet that is backing off. These two monopolization phenomena can provide a possible explanation for the peculiar shape of the throughput curve, which shows an initial sharp decrease (as packet monopolization becomes ineffective), followed by a steady increase as the number of frames in the system increases further and stations begin exhibiting channel monopolization.

Another observation derived from the response time and throughput curves is that "overcrossings" of these curves in the region $n \in [1, 130]$ indicate that systems composed of more workstations perform better as opposed to the expected behavior (i.e., the more the workstations, the worse the performance). This could be an artifact of the backoff algorithm, since when several frames queue up at a single station, only the one currently at the head of the transmit queue is considered for transmission over the channel. If that head frame is delayed for a long time due to experiencing a large number of collisions, then the remaining frames are delayed for a long time too. In particular for the case of two stations, whenever one of them experiences a particularly long backoff delay, most of the frames in the system tend to pile up behind the delayed packet, which amplifies the

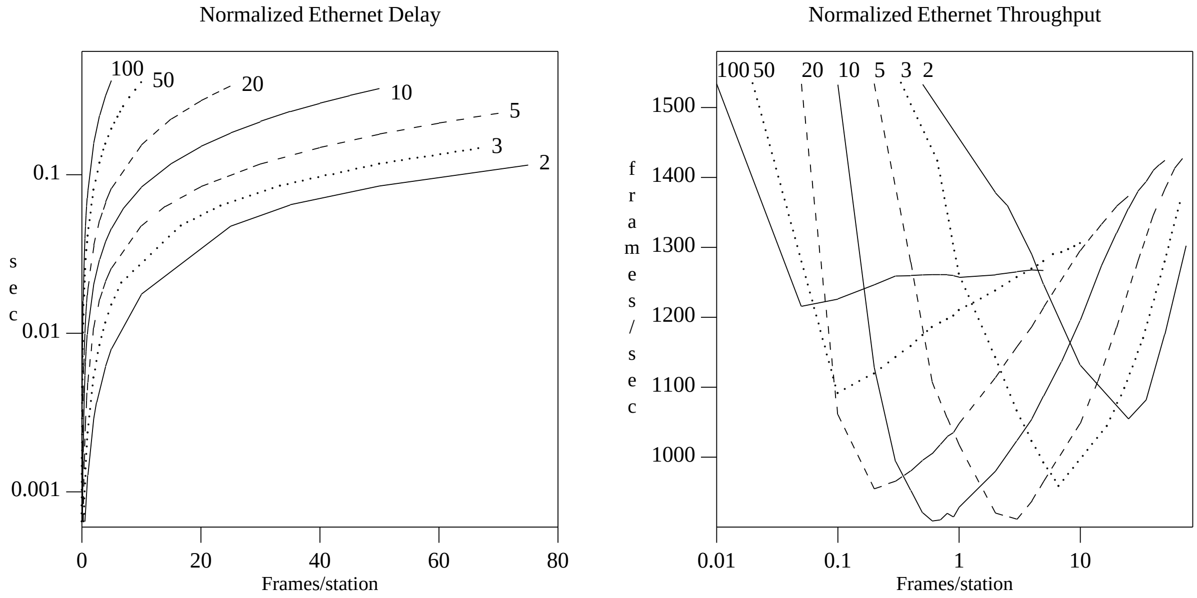


Figure 4.10: Normalized Throughput and Response Time

effect of the backoff delay on the per-packet statistics. Thus systems with less traffic per workstation appear to have better performance under moderate load, than those with more traffic per workstation. This result may have some negative implications for the response time of client/server systems, network bridges, etc., which are busy enough to experience collisions but not busy enough to take advantage of channel monopolization.

Consider that when there are m stations trying to share the Ethernet, on average the service time as seen by a particular station (i.e., elapsed time that a frame spends at the head of the local transmit queue) increases linearly with m, whereas under symmetric routing the arrival rate to a single station decreases linearly with m. This means that the utilization of the Ethernet as a whole and the apparent utilization of the Ethernet as seen by each station are the same, which for the case of an *open queuing system* would mean that the mean queue length at each station would be independent of the total number of stations in the system. Because of this observation, one might argue that the "fair" way to compare systems with different numbers of workstations, is to equate the average number of frames per station and not the total number of frames in the system. This even makes sense when we think about a real application, where each workstation would bring with it some inherent demand for the Ethernet, so the load on the system (i.e., frames wishing to be delivered at a given instant of time) ought to be proportional to the number of workstations.

For this reason we attempted a normalization of both response time and throughput which would enable the comparison of systems composed of different numbers of workstations based on equating the number of frames-per-station for all the cases. Thus we replaced the units on the X-axis so that the unit was frames-per-station rather than frames-in-system. The results are shown in Figure 4.10. The normalized delay clearly showed that systems with more workstations experience higher delays. However this was not the case for the throughput curves which did not get rid of the previously mentioned overcrossings. Interestingly, the minima in the throughput curve are clustered in the neighborhood of 1 frame per station, which is close to the operating point for a typical network intensive application.

4.3.2 Performance of "modified" Ethernet

Besides the traditional Ethernet protocol we have examined a variation of it suggested by Melas and Molle [47]. This paper describes an analytic study of the Ethernet performance where bulk data transfers are taking place. It makes the assumption of packet-train arrivals at each station. These packet-trains have an aggregate Poisson arrival rate G and their lengths are geometrically distributed. The analysis is based on an embedded Markov chain and applies to cases where "back-to-back" packets are separated by at most a round-trip propagation time (2τ). The model predicts that the system performance degrades significantly as the space between successive packets in the train approaches the 9.6 μsec gap imposed by the specifications. In fact sending the entire packet train as a single "monster" packet is shown to improve the performance significantly.

However the above analytical model did not consider the case where the interframe spacing exceeds the round-trip propagation time, because this condition complicates the analysis to a large extent. Therefore it does not provide with an "optimal" spacing that minimizes the interference among packet trains. The effect of a pause such as $2\tau + \epsilon$ (where ϵ is an infinitesimal time slice), would be the prevention of the channel monopolization . In other words a "distinguished" station (i.e., one that has just transmitted a packet successfully) must wait for time $2\tau + \epsilon$ before continuing, thus enabling other stations that are ready to send deferring packets to do so, as soon as the channel turns idle. Such "deferring" stations will sense the channel idle time τ after completion of the successful transmission by the "distinguished" station and start sending their packets. Time τ later the leading edge of this signal will reach the distinguished station which will remain inactive for ϵ time units more, and will defer to the new packet transmission.

Evidently for the case of two stations the above scheme results in optimum performance, since the stations are synchronized so as to interleave packet transmissions and avoid any interference. Nevertheless, this intuitive explanation cannot be applied to larger systems and since analysis is intricate, our simulation model can be properly instrumented to predict the effect of the "monopolization prevention" gap. The results are shown in Figure 4.11.

It is clear that in the case of 2 workstations both the delay and the throughput are significantly improved. The throughput remains practically constant throughout the test

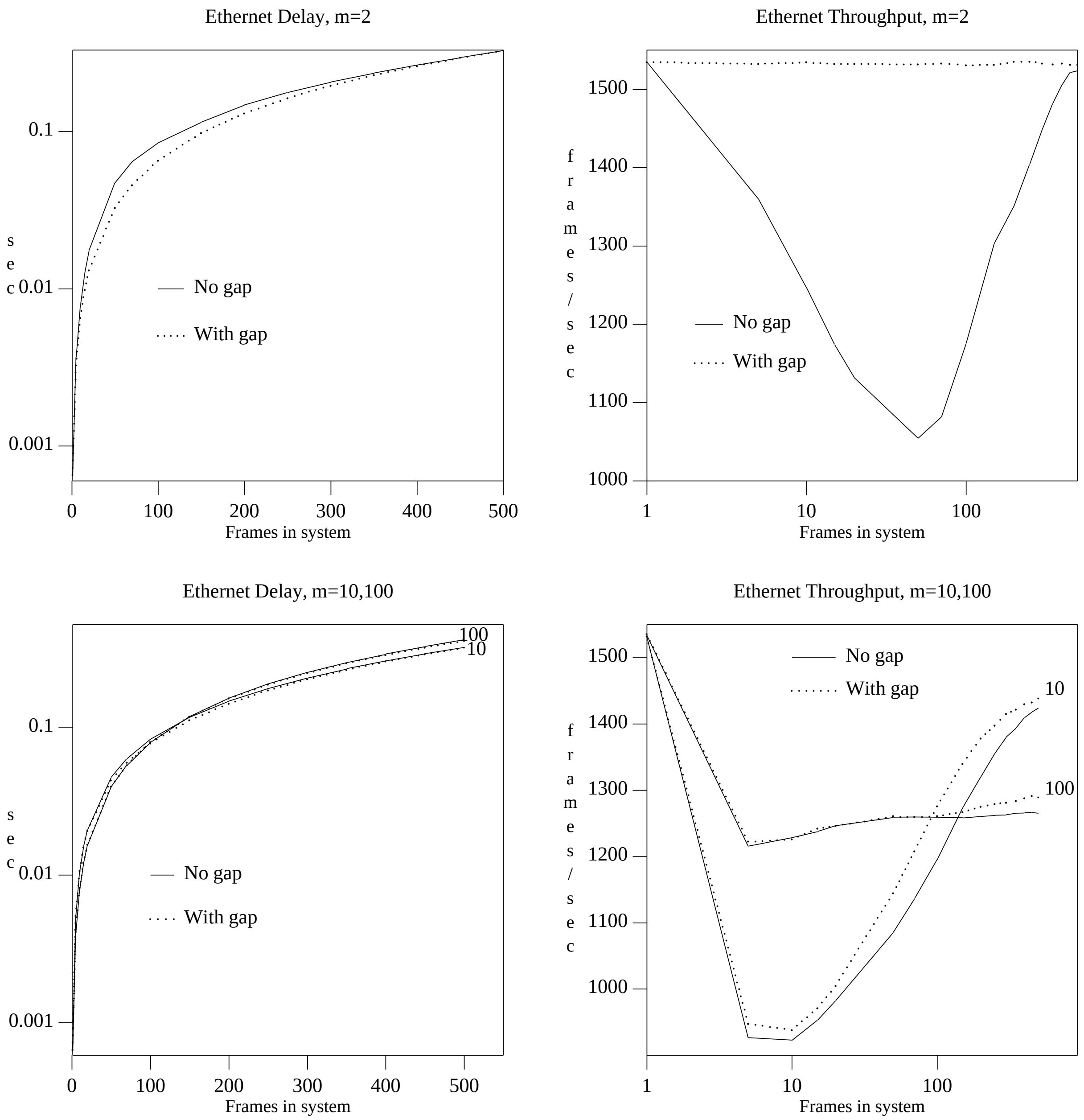


Figure 4.11: Gap vs. No gap version

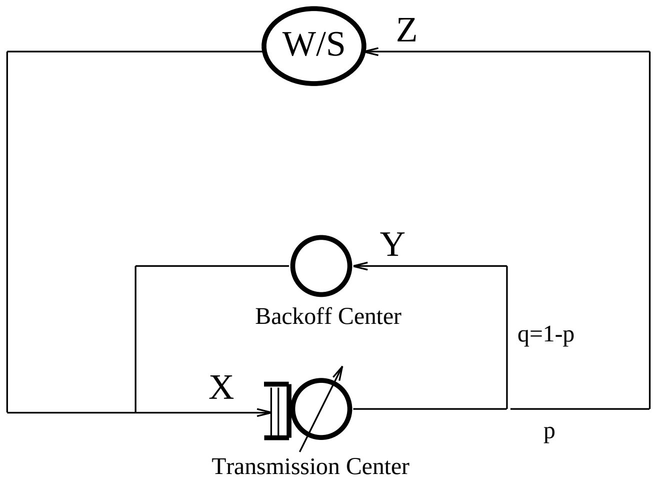


Figure 4.12: Sketch of a modeling approach of the Ethernet subsystem

regardless of the number of frames in the system, since collisions are totally avoided. However enhanced performance is also evident for systems up to 100 workstations, although the improvement for such large systems was only a 2-5 % improvement in throughput. Thus the application of the $2\tau+\epsilon$ gap provides a simple way to better system performance in the limiting case of high traffic loads generated by high speed processors and efficient transfer protocols. This conclusion is reinforced by the fact that a further increase in the inter-packet gap does not enhance the network performance. Indeed, we have tried several cases where the value of the gap is in the order of multiples of $2\tau+\epsilon$, via simulation. In all these tests the performance deteriorates as the gap increases above the value $2\tau+\epsilon$. Likewise, simulation and analysis can prove that the performance degrades as the gap decreases under $2\tau+\epsilon$.

4.3.3 Analytical Modeling Considerations

In this section we emphasize in the comparison of the previous simulation results to analytic ones derived by MVA modeling techniques. The closed queuing network for the above system is simple and consists of m queueing centers (each representing a workstation) and one FESC that represents the Ethernet subsystem. The workload comprises a single class of customers whose population is equal to the number of requests, n, residing in the system. The service time at the queuing centers is the processing time, D, at the workstations, whereas the processing time at the network load dependent center is determined by the equation (2.4). The visit count to each station center and the network center is $1/m$ and 1 respectively,

Generally the system under consideration obeys the separability constraints except

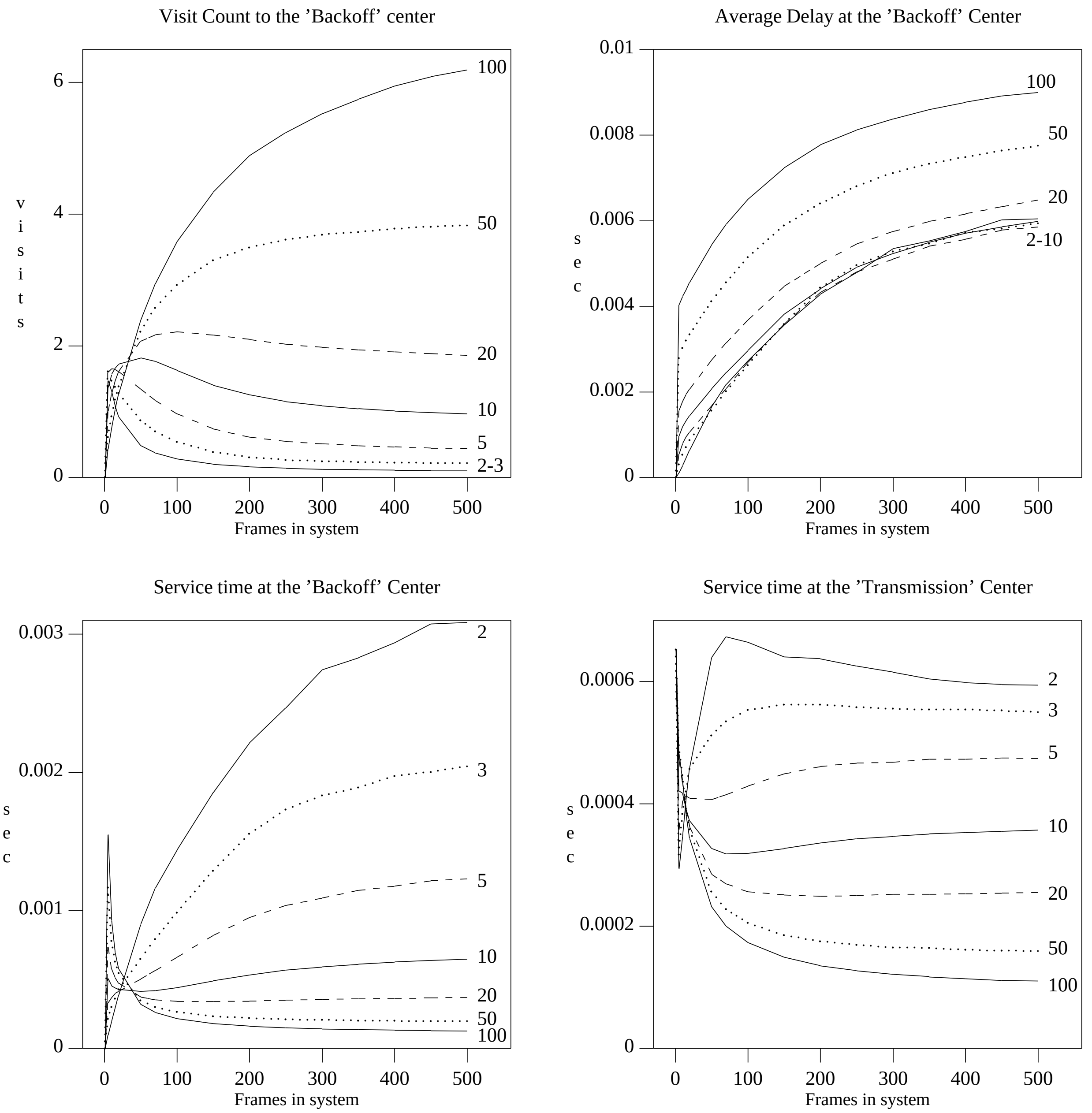


Figure 4.13: Additional Ethernet subsystem measurements (D = 0)

from the service time in the workstations which is deterministic, violating the assumption of exponential service times at queueing centers where the queueing discipline is FCFS. Fortunately this problem can easily be solved by employing a modification of the residence time equation of the exact MVA algorithm. The modification applies to cases where the coefficient of variation of the service time at a particular FCFS center is not equal to 1, which corresponds to exponential service times ([61]). For the particular case of constant service times and a single customer class, the residence time equation for these centers becomes:

$$R(N) = V_k \cdot S_k[1 + Q_k(N-1) - \frac{U_k(N-1)}{2}] \tag{4.6}$$

where k denotes the center, whereas N the population level, and V_k, S_k, Q_k, U_k are the visit count, service time, queue length and utilization of the center k. Half of the center utilization is subtracted from the time-averaged queue length at each iteration of the algorithm, to account for the residual life of customers in service.

For the case of no processing at the workstations the Ethernet FECS represents the entire system operation. As we mentioned in section 4.2 the existing Ethernet model does not distinguish between systems composed of different numbers of workstations and yields a single curve for all configurations. From Figure 4.8 we can conclude that the model performs poorly and is not able to capture the observed abrupt drop in the throughput. In terms of response time, the relative error ranges between 10 % and 20 %.

The natural approach that we could follow in order to derive a more accurate analytic model is to divide the network subsystem into two centers as it is shown in Figure 4.12. The *transmission* center holds the frames that are competing, whereas the *backoff* center corresponds to frames that are backing-off because they have experienced collisions. Intuitively the transmission center should be a load dependent one (with service time depending on the number of competing frames) whereas the backoff center should be a delay center introducing a delay derived by the retransmission algorithm. The idea behind this structure is to provide a better estimate on the number of frames that actually participate in the contest for the channel acquisition. This is the major weakness of the existing Almes and Lazowska model, which the above scheme would possibly capture by evaluating the number of frames that are backing off and therefore do not strive for the channel access.

In Figure 4.12, p denotes the probability of success per attempt averaged over all stations. The visit counts V_t and V_b at the transmission and the backoff centers respectively can be easily evaluated by taking into account the following equations:

$$\mathbf{X} = \tfrac{1}{p} \cdot \mathbf{Z} \qquad \text{and} \qquad \mathbf{Y} = q \cdot \mathbf{X} = \tfrac{q}{p} \cdot \mathbf{Z}$$

where $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ denote the in-flow at the transmission and the backoff center as well as the workstation subsystem respectively. The above equations imply that $V_t = 1/p$ and $V_b = q/p = V_t - 1$.

Instrumenting the simulator properly we can estimate the above parameters as well as the mean delay at the backoff and transmission centers and therefore evaluate the average population at both centers at various system total population levels. Thus we can work out the exact MVA algorithm in reverse order to define the service times at these centers. We investigated the existence of patterns that these service times might follow. Figure 4.13 shows several of these measurements.

V_b increases with the number of frames in the system, for $m = 50$ or 100 workstations. However if the system consists of 20 workstations or less, V_b decreases as the number of frames increases. This is an artifact of the fact that Ethernet's backoff algorithm tends to shrink the contention part of an idle-contend-transmit network cycle, thus increasing the duration of the idle sub-cycles and consequently the probability of successful transmission. The average delay at the backoff center appears to be practically the same for $m \leq 10$ stations but this is not the case for larger configurations. Finally the service time curves show that there is no general specific rule that governs their definition. Hence we were not able to define a simple queueing network of this form that holds for all combinations of the parameters.

The moral that can be extracted from this experiment is that although there is still room for an improved model that can predict the exact Ethernet behavior by means of a better encapsulation of the properties of its backoff mechanism, it has become clear that this detailed model will not be attractive in terms of complexity and ease of application.

4.3.4 A More Realistic Configuration

A presentation of the performance of a closed system where some amount of processing is involved, completes this section. This is a more realistic case since its is directly applicable to many real networking environments. We have chosen the case of $D = 5$ msec for which the ratio of the workstation service demand to the network service demand is approximately equal to 10. This reflects many typical applications. The simulation results are displayed in Figure 4.14.

We can see that the network becomes the system performance bottleneck when the number of interconnected stations is between 5 and 10. For configurations where the workstations are the bottleneck devices, the simulation and analysis are in remarkable agreement. This holds for all cases for which the choice of the parameters m and D leads to saturation of the workstations.

However this is not the case for configurations where the network is the bottleneck center. As we can see the analytic model gives almost the same result for all above systems which is usually an optimistic one. We can also see overcrossings of the throughput and delay curves in the region $n \in [1, 130]$. This can be explained by the same argument that we used in the analogous observation for the $D = 0$ case.

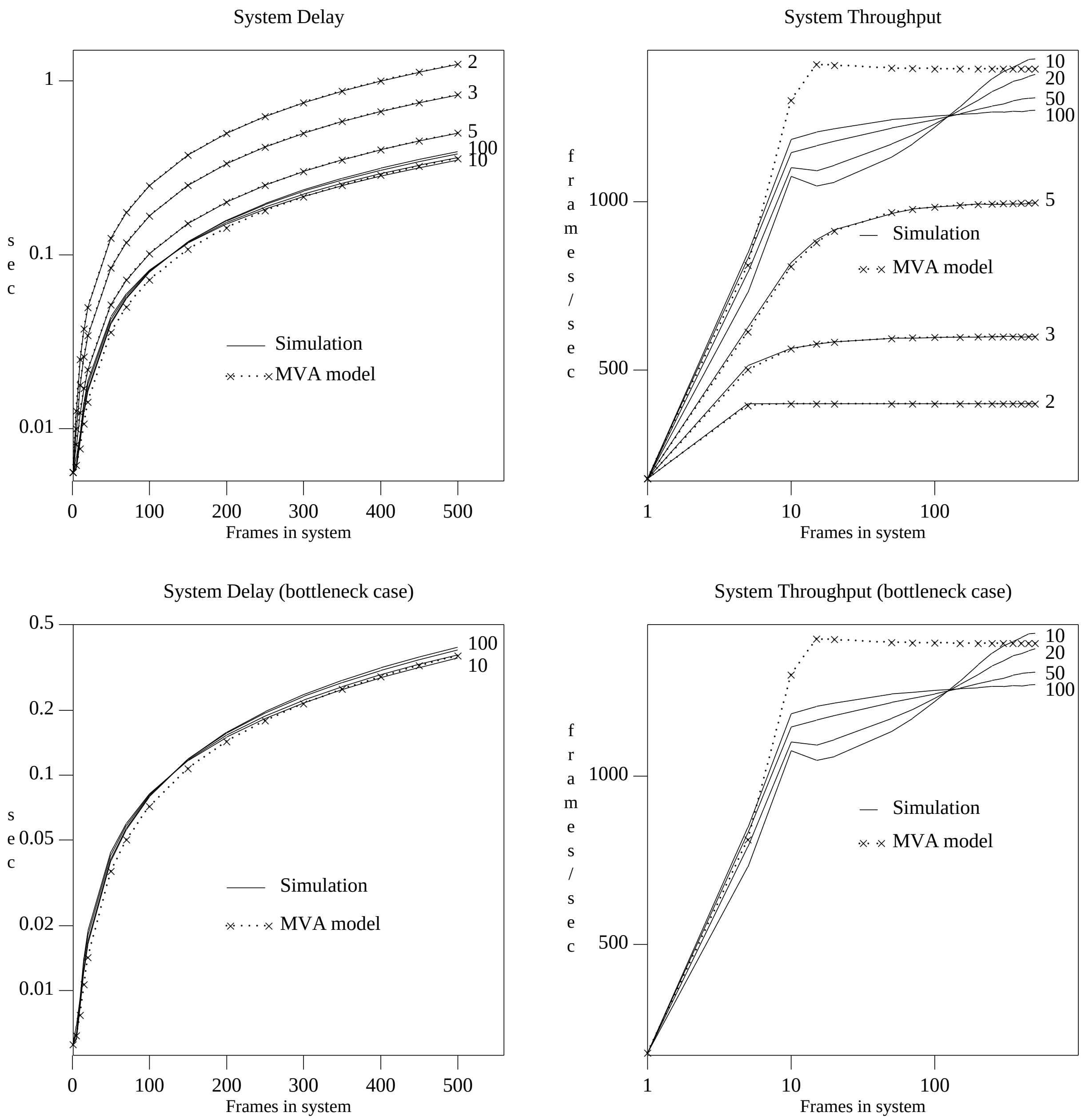


Figure 4.14: Ethernet Performance in a closed system with D = 5 msec

4.4 Ethernet in a Client/Server Virtual System

One would normally expect that the last part of our experimentation process would involve a real computing environment in which we should focus on the Ethernet performance. Such a study should be carried out with respect to a specific application running on the particular system, in order to investigate whether the Ethernet becomes a bottleneck center in this environment. Unfortunately we were not able to have such a system in our disposal, since besides the system itself, this required the existence of appropriate measurement tools for benchmarking and system tracing, so as to be able to define the system parameters (demands at the centers etc.), as well as to characterize the workload in regards to a specific application. Therefore we decided to experiment on a "Virtual" (Hypothetical) Client/Server architecture, influenced by distributed systems described in several articles in the literature, and stress-test it in terms of applying an intensive workload to the Ethernet.

We have presented (section 2.2) results from several studies that have been done in connection with certain systems. Our intention is to parameterize our simulation model using the system parameters described in some of those articles in order to examine the Ethernet behavior and derive our conclusions about its effect on the system performance. However, the system descriptions in those publications are characterized by minimal detail, and resort to tabular information regarding service demands at the various centers. There are very little data relevant to the network characteristics (topology, packet sizes, length, traffic intensity etc.) that would facilitate a detailed instrumentation of our simulation model. In several cases this can be a problem and lead to unrealistic results. To get by, we made an effort to construct a system without sophisticated elaboration emphasizing in the network contention factor (which is actually the issue in this research) and not in the performance of the operation of the other system devices (paging devices, Direct Memory Access, Communication Interfaces etc.). The system description and the illustration of the results follows.

4.4.1 Experiment Description

We based our investigation on a slight modification of the experimental configuration described by Boggs et al. [29] (the one we used for the validation experiments in section 4.1), using data derived from the HP system illustrated by Molloy [42]. The latter system wires up several HP 9000 workstations to a single file server, supported by the DUX (Distributed Unix) operating system that enables remote transactions (reads, writes, paging). We configured a client/server architecture where a number of *client* workstations were associated with a single *server* station. The system comprised of several such sets of client/server clusters. In each set multiple clients could have connections to the server (by sending packets of either constant or bimodally distributed length to it), and the server replied by addressing the packets to its clients each of which was picked out in a uniform man-

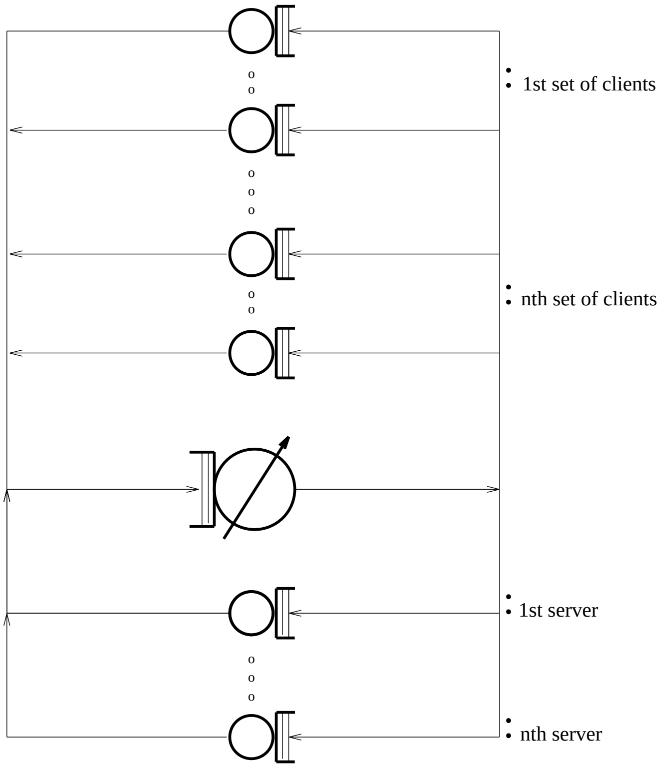


Figure 4.15: Model for a client/server system

ner, before every packet transmission (i.e., we are assuming an asynchronous client/server communication). Thus we defined a highly distributed computing environment (assuming only remote transactions) where we wished to examine what would happen when a large number of workstations started running applications simultaneously.

A comparison of the service demands of the various components of the architecture as described by Molloy [42] (local client CPU, remote server CPU and LAN) for a particular type of remote transaction, enabled us to determine the ratios r_c, r_s, of the client service demand and the server service demand to the network service demand respectively. For file writes we determined that the above two ratios were approximately 10 and 8 respectively. We used these values to set the demands of the service centers in our simulation.

Another issue that we wished to tackle was the effect of queuing in the system. Queuing delays can be caused by congestion at the shared resources (i.e., servers in cases that they are associated with multiple clients) or due to a highly loaded system. These delays introduce variability in the packet arrival process at the network and raise the average delays.

We have repeated experiments for several possible combinations of the following parameters:

- clients per server, φ, (e.g. 1, 4, 9),
- average number of packets per station, q, (e.g. 1, 5),
- packet lengths (e.g. 64 bytes, 512 bytes etc.), and
- packet length distribution (deterministic, bimodal).

In the following section we will present the results of runs with constant and bimodal packet sizes with the same average length. Particularly, we consider $P = 362.67$ bytes in the deterministic case and $P_s = 64$ bytes, $P_l = 512$ bytes, and $P_{bd} = 1/3$ in the bimodal case.

We should point out that the way we set the demands in our model, corresponds to a *stress-test* in the real system, because in reality it's rather unusual to find a system that is busy doing only file reads and writes. In an interactive environment the *think time* associated with each transaction plays a crucial role in the system load because it decreases the rate that requests are submitted to the system, hence the network can allow for more workstations to be attached to it. This brings up the problem that we mentioned in the beginning of the section about the level of detail with which the system under consideration was portrayed.

4.4.2 Simulation Results

An analytic study via use of QNMs would enable the comparison of the output of the above simulation with results obtained from the analytic point of view. A "general" model that can be used to describe such client/server architectures is shown in Figure 4.15. This closed queuing network consists of as many queuing centers as the total number of stations in the system, and a FESC[6] for the Ethernet representation. The workload comprises of a number of classes each of which corresponds to a set of clients and their associated server. The service time at each center is equal to the processing time at the particular center. The population of each class is equal to the total average number of customers (frames) in the clients and the server of each client/server set (e.g. for $q = 5$, $\varphi = 9$, the population of each class is: $5 \cdot (9+1) = 50$). For the customers of each class the visit count must be set to $1/\varphi$ for each client of the corresponding set, 1 for the server and the network, and to 0 for all other centers in the system.

The throughput and response time results for the case of no queuing (i.e., $q = 1$), and constant packet sizes are illustrated in Figure 4.16. These curves show the plot of the

[6]This FESC is the same as the one that we used for the Ethernet models in the previous sections. It is based on the Almes and Lazowska model.

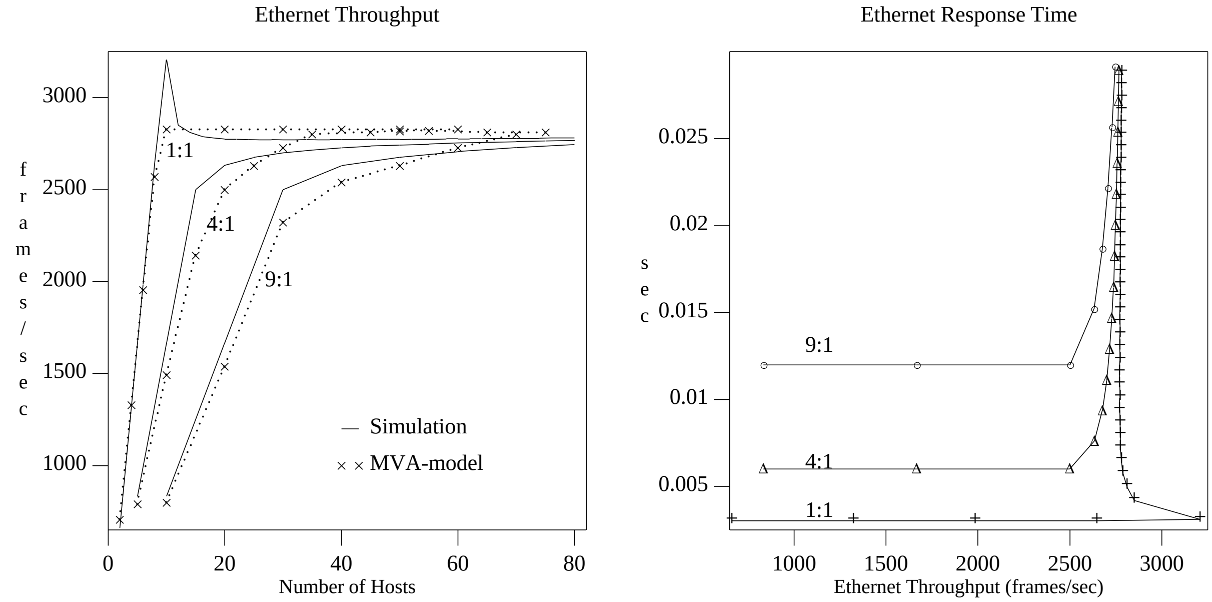


Figure 4.16: System performance for constant packet sizes, and no queueing

performance measures as a function of the total number of stations in the system, and they have been labelled by the client to server ratio ($\varphi : 1$). For the case of 1 frame per station the simulation study proved that the network started making contributions to the system response time at rather intermediate-intensity loads (determined by the number of workstations). The point at which the throughput leveled off (often called as the *knee* of the throughput curve) was different for the various (in respect to client to server ratio) configurations. For the 1:1 system, the response times began to inflate at 12 workstations whereas for the 4:1 system the corresponding point was at 15 workstations and for the 9:1 one, saturation phenomena appeared at 30 workstations. The 1:1 and 4:1 systems performed very closely after 16 stations whereas all configurations had similar behavior after 30 workstations. The reason for different performance in the region of 2-30 stations was caused by the different client to server ratios. When there was no substantial network overhead in the system delay, the 1:1 system outperformed the other two configurations, and the 5:1 system beat the 9:1 one, because of the lower delays at the server stations.

During saturation the Ethernet utilization ranged between 70%-75%, and 80 to 90 percent of the population in the system resided in its queue. The average number of attempts per frame transmission never exceeded 4. For the 4:1 system the ratio of server utilization to client utilization was approximately equal to 3.5, whereas for the 5:1 system this ratio was about 7.5. The standard deviation of the throughput (among the various stations) was initially very high for the 4:1 and 9:1 system due to the outrageous utilization of the servers in contrast with the clients utilizations, but it leveled off as the number of

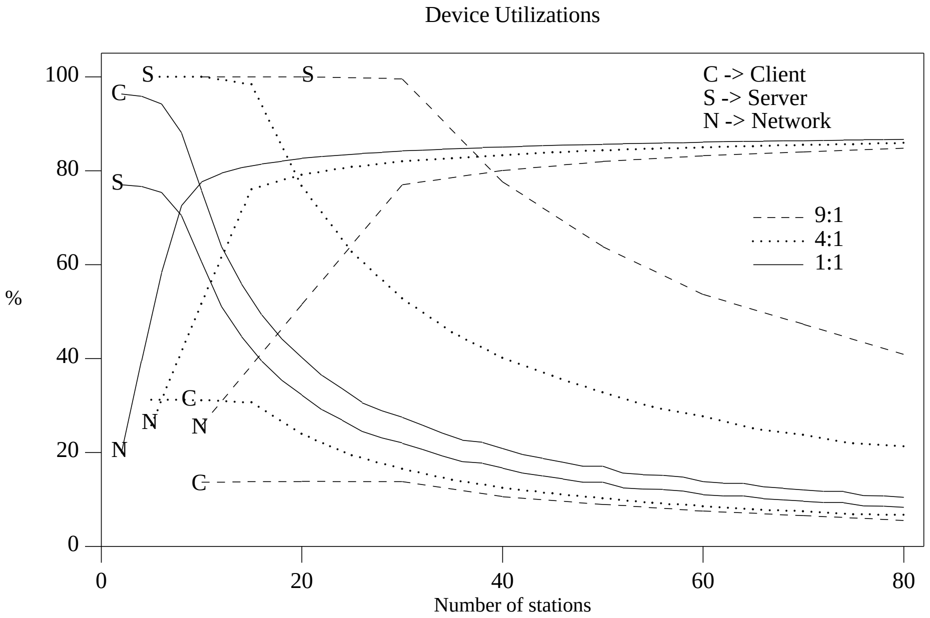


Figure 4.17: Utilization of centers for bimodal packet sizes, and no queueing

(client/server) clusters increased. The average number of frames sent in a row from a station was bounded by 2.

Finally, there is an interesting observation, regarding the throughput curve for the 1:1 system with constant packet size, in Figure 4.16. The throughput is continuously increasing with the number of workstations, peaks at 10 workstations and then drops dramatically for 12 workstations, giving a very steep slope in the curve. We discovered that this situation is an artifact of a cyclic behavior, consisting of perpetual repetition of a routing pattern. This pattern is composed of packet exchanges between the members of each pair of workstations. This pattern in formed due to the time t_s, between successive transmissions from the same server (which is the faster device between the client and the server stations in our system). According to the ratio server service demand to packet transmission time, 8 packet transmissions can fit in the interval t_s. But for the 10 workstation system this means that all other pairs (namely 4) of workstations can interchange packets (8 transmissions) before a "distinguished" server sends the next reply to its client. Therefore, a routing pattern that involves minimal collisions can be established yielding a peak in the system throughput. Note that for more workstations this pattern will eventually start to decay because the t_s cannot allow for more packet interchanges and the collision ratio will increase deflating the system throughput.

The utilization of the various system components can be seen in Figure 4.17 for several values of the parameter φ. We can clearly see that initially the servers are saturated (except for the case of pairs of workstations that interchange packets), but as we include

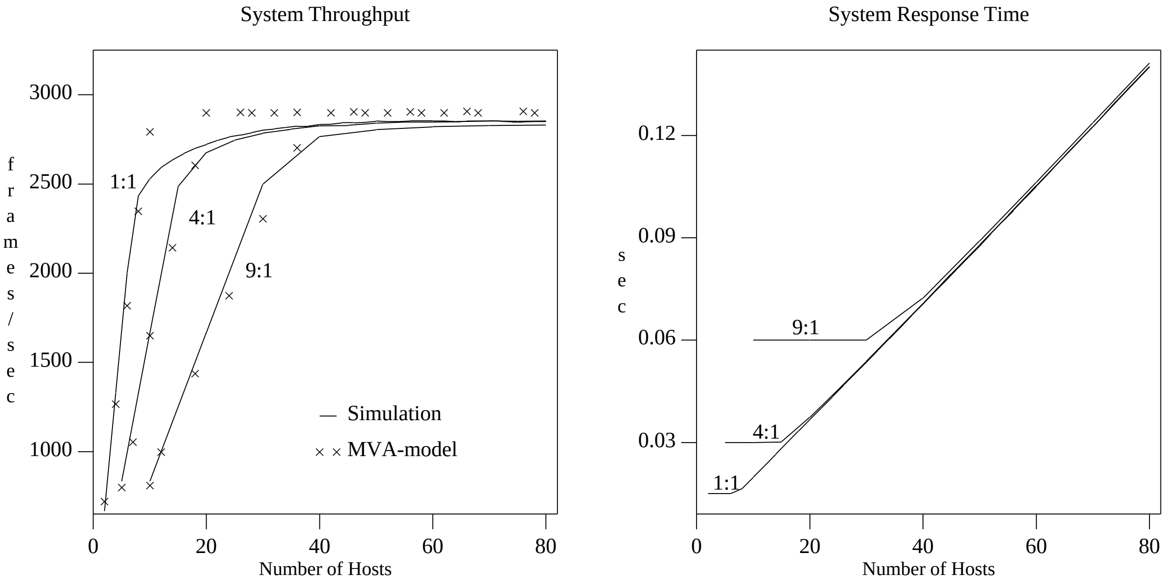


Figure 4.18: System performance for bimodal packet sizes, and queueing

more sets of client/server clusters and increase the volume of traffic on the Ethernet segment, the network saturates and the utilization of the client and server stations degrades substantially. Particularly, we can see that a 40 workstation system with 4 servers, keeps the network 80% utilized, the servers 70% utilized, and the clients 25% utilized.

The effect of queuing was quantitatively remarkable, as we can see in Figure 4.18. Qualitatively it affected the performance of the 1:1 system so that it saturated earlier than before (6 instead of 10 stations), and eliminated the differences in performance between all three configurations after the point that all of them saturated the network. This came naturally after the higher utilization that was observed at all centers before and after network saturation.

In this case the bit rates appeared to be slightly higher and the response time approximately 5 times higher than the case of no queuing. (Note that here we had 5 frames per station on average.) The concentration of customers at the Ethernet queue, the utilization of it, as well as the collision rates were higher. Furthermore the monopolization phenomenon was more obvious since the mean number of frames sent from a station exceeded 2 and had a variance of as high as 5. This can possibly be attributed to the relatively higher concentration of frames at the server station.

Finally, the case of bimodally distributed packet lengths didn't have substantial quantitative differences with the constant packet size case. Its effect was basically the elimination of "peaks" in the throughput curves. These peculiar "peaks" in the case of $\varphi = 1$ systems is an artifact of a synchronization effect which we will describe at the end of the current section. As in the previous case, queuing inflated the response time, increased the device utilizations and stressed the monopolization phenomenon.

Earlier studies (e.g. [41]) claim that Ethernet makes no substantial contribution to response times. Comparing in Figure 4.18, the point where response time takes off with the point that throughput levels off, we see that there is a region (namely between 8 and 25 workstations) where the Ethernet contributes to the system response time before saturation, at least for the 1:1 curve. Thus, we have found that the Ethernet can play an important role in the system performance in contrast with earlier Ethernet studies when the speed of the workstations wasn't capable of leading to a network overflow. The older claims that a single Ethernet can reliably accommodate several servers and their associated workstations before network contention becomes a serious issue, are no longer in effect.

A limiting study of client/server systems based on realistic workloads can prove that in many cases, Ethernet plays a great role in the calculation of the response times and should not be ignored. The simple experiments that we carried out, were based on a realistic system and an – although intense – but manageable workload. Therefore our results reinforced the implication of the Ethernet congestion.

As we can see in Figures 4.16 and 4.18, the analytic model outputs are in reasonably good agreement with the simulation results. This MVA model, however, is not a very attractive one, due to the large number of centers and classes that it incorporates. Its solution has been done via use of the MAP performance evaluation package and and it is based on approximate MVA methods. Exact solution, increases the computational cost to a very large extent. We can see that the model gives an optimistic result for the region of network saturation revealing the inaccuracy of the existing Ethernet analytic model at high loads. Finally, we observe that the peak in the throughput curve in Figure 4.16 cannot be captured by the above model, which is no harm since this type of synchronization is unlikely to happen in real systems.

Chapter 5

Experiments on VT-CSMA/CD-Based Systems

In this chapter we have developed an accurate simulation model of the asynchronous Virtual-Time CSMA with Collision Detection (VT-CSMA/CD) protocol that was described in section 2.1.2. We have instrumented this simulator appropriately, so as to carry out a series of tests in the same distributed environments as the ones in which we tested Ethernet, in order to compare the behavior of the two protocols. Again, we focus on the closed systems context and derive a simple analytic model that accurately represents the VT-CSMA/CD operation in this case. But before that we illustrate a few details of the implementation of the model and the choice of the system parameters according to the theoretical studies relevant to the VT-CSMA media access algorithm.

5.1 VT-CSMA/CD Simulation Model

A detailed implementation of the asynchronous non-priority and priority versions of the VT-CSMA protocol, based on a parallel simulation model implemented in a programming language with concurrency, is presented in [21]. Each station is represented by an independent process that executes code similar to what a real station in a VT-CSMA network would use. Using this approach, one can construct a very realistic simulator. However, it is not very efficient in terms of execution time, and exceedingly difficult to debug.

In our model we simulate the operation of all stations by a centralized event-driven structure which is an extension of the model presented in chapter 3 for the Ethernet LAN. We have used the same events but we have modified the event routines to take into account the more complex scheduling mechanism of the VT-CSMA/CD protocol. Ethernet transmits "ready" packets on an idle channel with probability 1. Packets that arrived during a busy period are deferring transmission until the channel turns idle. If at least two new packet arrivals were experienced before the end of the busy period a collision is guaranteed.

On the other hand, VT-CSMA/CD provides a scheme to "spread" such arrivals out in time, in order to minimize the probability of interference among them. This mechanism is based on the existence of two clocks as we described above. Each station's virtual clock is simulated by means of two variables. The first one keeps the last update of the value of the virtual clock, whereas the second one keeps the real clock reading at the time that the virtual clock was last modified. This information, along with the virtual clock rate and the packet timestamps, allow us to estimate the (real) time that each packet in the transmit queue of the station will be transmitted.

Every time that the "originator" station changes the channel state to IDLE[1], all stations recalculate the (real) time that the first packet in their transmit queue should be transmitted. The transmission time of each packet is kept as a separate field in the packet record. The time of the next actual transmission is then taken as the minimum of these scheduled transmission times. Since the transmission time of the packet at the head of the transmit queue must be updated each time the channel is sensed busy, we do not insert the ATTEMPT events in the event list for every station, since typically only a few would actually get executed and the rest canceled. Instead we are using the following scheme which alleviates the complicated manipulation of the event list.

We use a buffer (which we call "head" buffer) to hold the packets at the head of the transmit queue of all stations. This buffer is always sorted with respect to the transmission time field of each packet. Every time that the originator station senses the channel idle, the transmission times for all packets in the "head" buffer are sorted.[2] Before we issue the ATTEMPT event for the first packet in the "head" buffer we keep track of the ARRIVE events in the event list, so as to ensure that no other packet arrival will take place prior to the ATTEMPT event for the first packet in the "head" buffer, which is scheduled to be transmitted closest in time. This is because if a packet arrival at a station with an empty transmit queue occurs, and the transmission time of the new packet is before the time of the ATTEMPT event of the first packet in the "head" buffer, then the newly arrived packet will be transmitted first and all other packets in the "head" buffer should be rescheduled for transmission, because their transmission time may change.

If no packet arrivals occur until the transmission time of the first packet of the "head" buffer, we insert the event ATTEMPT for this packet. ATTEMPT events will also be issued for all packets in the "head" buffer whose transmission time is within τ time from

[1]The channel state names and event types are the same as in chapter 3.

[2]Complete sorting of this buffer is not always necessary. It is usually done by means of insertion sort where a new element is out of order and should be inserted in a sorted array. However, in cases when complete sorting of the buffer is required we apply either bubble-sort or quick-sort ([62]). The choice is based upon the number of elements in the array. If this number is greater than 9 we pick quick sort to optimize this part of the process.

the transmission time of the first packet.[3] If such packets exist, the system will experience a collision which will be resolved by a backoff algorithm. In this system, for compatibility with our earlier Ethernet experiments, we assume the Ethernet's truncated binary exponential backoff algorithm. The choice of the retransmission algorithm is not very important since the backoff algorithm does not affect the system performance to a large extent as we mentioned before.

The virtual clocks for each station are updated every time the channel state changes from IDLE to either XMISSION or COLLISION and from either XMISSION or JAM to IDLE. The calculation of the scheduled transmission times must take place every time the channel becomes IDLE, after either a successful packet transmission or a collision subcycle. If a collision occurs, a random backoff delay is added to the packet timestamp and the packet is reinserted in the transmit queue so as to maintain the ordering of the queue with respect to the packets timestamps. The sorting of the transmit queue means that each packet is treated independently by the protocol. Subsequent packets in the same transmit queue can jump over a packet that experiences a long backoff, as opposed to the Ethernet policy where those packets can never been considered for transmission before the head packet is successfully transmitted. Thus, the performance of VT-CSMA is less sensitive to the distribution of traffic over the stations. In particular, it should avoid the "starvation" problem we observed in Ethernet under moderate load, shown in Figure 4.4, and we expect a reduction in mean message delay in this case. In almost all cases, this reordering policy performs most satisfactorily. However, we noticed that there is one case when this policy created some problems in the initialization phase of the system which corresponds to bulk packet arrivals at the stations. In our case we encountered the problem in the context of large populations in the closed systems, where we observed massive collisions and high error rates during system initialization.

For those systems we applied a modification of the protocol which solved this bulk collisions problem. According to this modification, in case of a collision, we stop the virtual clocks at the colliding station until time equal to the duration of the backoff interval of each station elapses and send the previously colliding packets again, immediately afterwards. Thus there is no need to sort the transmit queues of the stations because we do not change the timestamps of the packets. This method proved to be very effective and efficient in terms of system performance and we will illustrate it in more detail in a later section in this chapter.

In both cases of open and closed systems, the parameters of the system at the workstations level and at the network level remain the same as in the case of the Ethernet model.

[3]This implies that we again maintain the assumption of the star network topology, which although reflects a "worst-case" network since all stations are separated by the maximum propagation time, it optimizes the complexity of the simulation. Other assumptions that apply to our model, as in the case of the Ethernet, are (i) infinite buffer space and (ii) instantaneous information exchange between the modules of a station.

An additional parameter vis-a-vis the network operation is the rate, η, of the virtual clock which affects the network performance. We usually set this parameter to the value for which the channel attains maximum throughput (capacity). Finding the optimal value for η is the subject of the following section. Finally we note that the 1-persistent CSMA protocol should appear as a special case of VT-CSMA for $\eta \to \infty$, so we tried some tests with $\eta >> 1$ to see that this is the case.

5.2 Protocol Capacity

In this section we will present a few elements on the theoretical analysis of the protocol that are useful to determine the choice of the parameter η in our simulation model. According to the notation used in the analysis and assuming a star network topology, both the end-to-end propagation time, τ, and the jam time j have been normalized to the packet transmission time and are now signified by a and c respectively (i.e. $a = \tau/P/C$ and $c = j/P/C$). Analysis also makes the simplifying assumption of Poisson total traffic with intensity (load) G per unit time. This traffic consists of new message arrivals and retransmissions. If we consider Poisson arrivals, the above assumption for G is very realistic due to the rare retransmissions, as opposed to the Ethernet where the backoff algorithm introduces serious distortions in the Poisson input process as we found in the previous chapter.

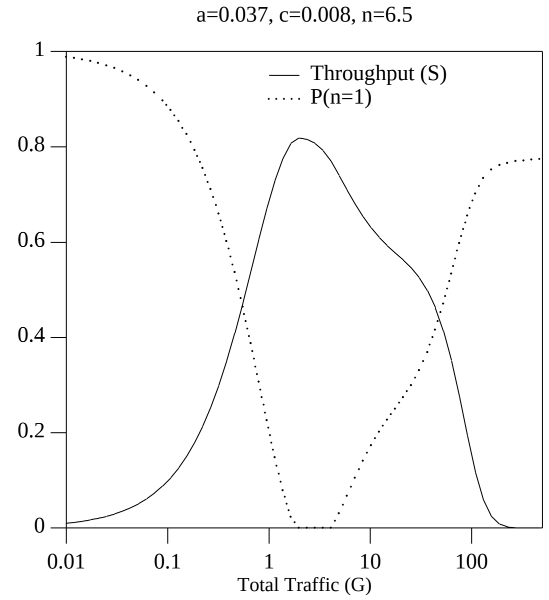
The virtual time protocol has two modes of operation, denoted by the virtual clock speed r. These modes are (i) "backlogged" (when $r = \eta$) and (ii) "caught-up" (when $r = 1$). Each mode can be viewed as an instance of the non-persistent CSMA protocol, each with different traffic values.

The throughput analysis of the protocol has been carried out by Molle and Kleinrock [5]. The channel activity has been partitioned into transmission cycles each of which runs from the beginning of the idle period that follows a successful transmission, until the end of the next successful transmission. Throughput is defined to be the ratio of the amount of useful work, $E[H|r]$, that is carried out during a random transmission cycle, to the expected duration of this cycle $E[L|r]$. Because of the two modes of operation the above entities have been conditioned to the value of r. Assuming asynchronous operation and collision detection these entities are given by:

$$E[H|r] = e^{-arG} \tag{5.1}$$

$$E[L|r] = c + 2a + (2 - e^{-arG})/(rG) + e^{-arG} \cdot (1 - 2a - c) \tag{5.2}$$

We also need to evaluate the equilibrium probabilities $p_{r=1}$ and $p_{r=\eta}$ that the system is in each of the above two modes of operation (i.e. caught-up and backlogged respectively). The evaluation is based on the observation that the average advance of virtual time during



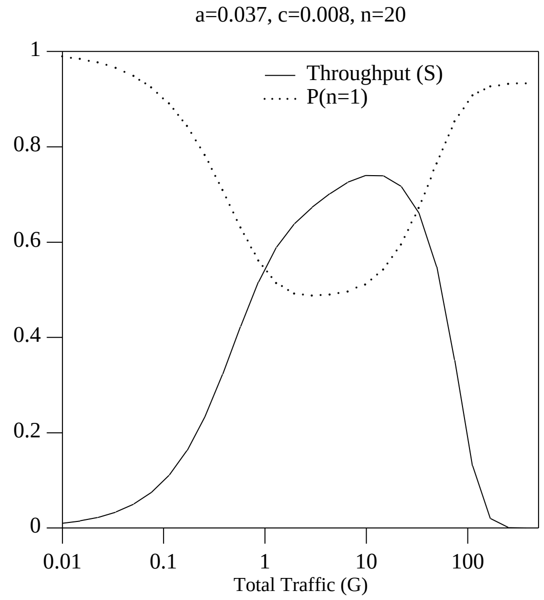


Figure 5.1: Throughput (S) and $p_{\eta=1}$ vs. Total Traffic (G)

a transmission cycle (i.e., $p_{r=1} \cdot (1/G + a) + p_{r=\eta}(1/G + \eta a)$) should be equal to the average length of the cycle (i.e., $p_{r=1} \cdot E[L|1] + p_{r=\eta} \cdot E[L|\eta]$), and yields:

$$p_{r=1} = min\{0, \frac{1/G + \eta a - E[L|\eta]}{(\eta - 1)a + E[L|1] - E[L|\eta]}\} \tag{5.3}$$

and $p_{r=\eta} = 1 - p_{r=1}$. Finally, the throughput S, can be expressed as:

$$S = \frac{p_{r=1} \cdot E[H|1] + p_{r=\eta} \cdot E[H|\eta]}{p_{r=1} \cdot E[L|1] + p_{r=\eta} \cdot E[L|\eta]} \tag{5.4}$$

For a given network (i.e., the values of a and c are known) and a given value of η we can plot the throughput S as a function of G. Figure 5.1 illustrates the throughput and the value of $p_{r=1}$ for two values of η on the same network.

In the same paper [5], Molle evaluates the capacity of the protocol which is defined as the maximum value of the throughput equation for which the expected message delay is finite. Assuming a stable backoff algorithm, the above provision is satisfied when the backlog never becomes asymptotically infinite (i.e., $p_{r=1} > 0$ or $p_{r=\eta} < 1$). Thereby, the capacity of the protocol is evaluated by determining the maximum among the values of throughput equation for which $p_{r=1} > 0$. Figure 5.2 shows the plot of the capacity of the protocol as a function of η for several values of a and c. The combinations of a and c that we picked reflect boundary values derived by the minimum and maximum values of the parameters τ, j, P according to the Ethernet specifications. Since no minimum network

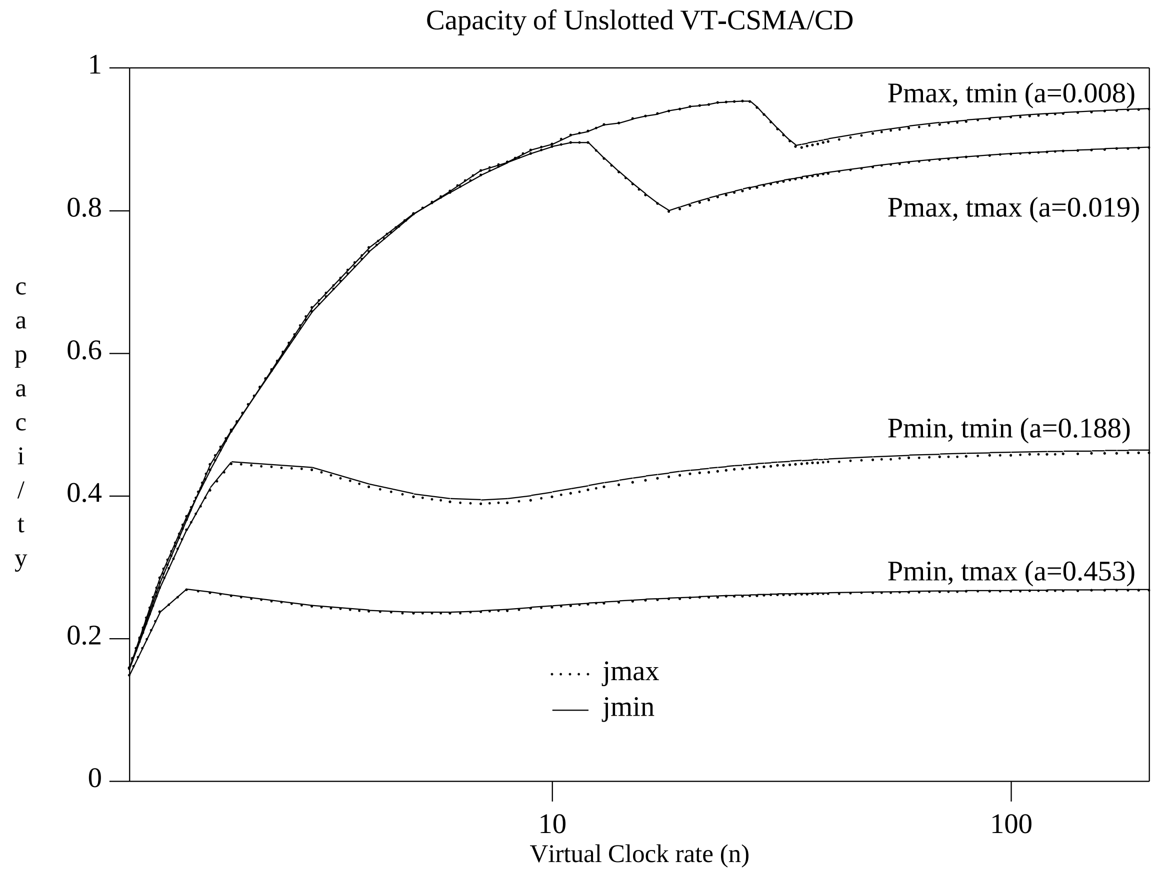


Figure 5.2: Capacity of Unslotted VT-CSMA/CD, for boundary values of P, τ, j

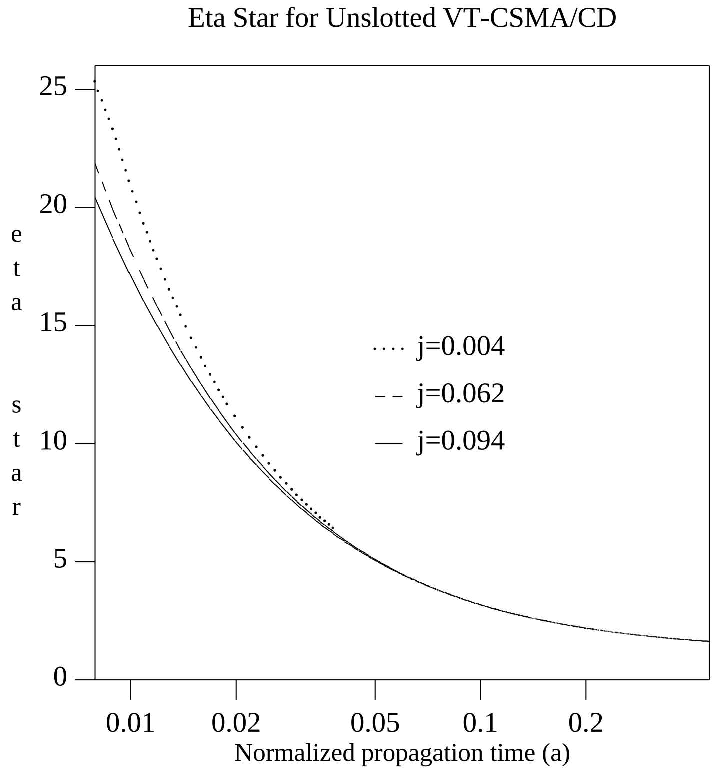


Figure 5.3: η^* versus a

length is specified we have used the value 9.6 μsec for τ_{min}. Thus, the above boundary values are:

- $P \in [64, 1512]$ (bytes)
- $\tau \in [9.6, 23.2]$ (μsec)
- $j \in [3.2, 4.8]$ (μsec)

It can be seen from Figure 5.2 that for a particular value of a, the capacity is insensitive to the normalized collision recovery time c, whereas it is highly dependent on a and η. Particularly for η there is a short region in which the capacity makes a noticeable drop especially for large values of a.

For certain a and c, the capacity has a local maximum for a particular value of η which is termed η^*. It can be proven that for this value of η, the capacity of VT-CSMA is the same as the capacity of the non-persistent CSMA, which is the best possible for interference-avoidance protocols under the Poisson total traffic assumption [4]. Let G° be the offered load for which the throughput obtains its highest value S° in the non-persistent case. VT-CSMA/CD will obtain the same value when

$$G \equiv G^* = G^\circ/\eta \tag{5.5}$$

But when $G \to G^*$, the probability $p_{r=1}$ approaches its asymptotic value which is zero, and by substituting G by G^* in the numerator of (5.3) we have: $1/G^* + \eta a = E[L|\eta]$. Using (5.5) and solving for η we get:

$$\eta = \frac{E[L|\eta]}{1/G^\circ + a} \tag{5.6}$$

The above value of η (η^*) is the one for which the capacity of the VT-CSMA/CD protocol approaches the capacity of the non-persistent one, which is the maximum possible. Figure 5.3 shows the η^* as a function of a for various values of c. It is shown that η^* does not depend highly on c, but it does depend on a. Small values of a correspond to high values of η^*.

5.3 Validation of the Simulation Model

First, we have attempted to validate the simulation model which we briefly described in the previous section. Validation is based on the analytic model described by Molle [6]. This paper extends the throughput and delay analysis that was initially carried out by Molle and Kleinrock [5]. The problem with the latter study is that it derived bounds for the delay which proved to be very weak for the unslotted version of the VT-CSMA protocol. This problem has been overcome in the follow-up paper, where a closed-form formula for the initial waiting time up to the first transmission attempt is derived, using a complex analytical methodology based on the Moving Server Queuing System model. Although this analysis pertains to prioritized environments, where several classes of messages each with different priority levels exist, the results are applicable to the special case of a single priority class in the system.

We have tried to instrument our simulation model so as to represent a system whose parameters match the ones reported in the analysis of Molle [6]. We consider a star network which consisted of m equidistant workstations. Every pair of stations is time $\tau = 23.2\mu$sec apart. The duration of the jam signal is $j = 2.32\mu$sec. The input traffic is Poisson and its intensity is bounded by the theoretical capacity of the protocol. Packets are of constant length, $P = 2900$ bytes. Assuming a 10 Mbps passive broadcasting medium the parameters a and c are equal to 0.01 and 0.001 respectively. We let the virtual clock run at rate $\eta = 10$ when the system is backlogged. For these values of a, c and η, the throughput formula of the previous section yields a capacity of 0.89.

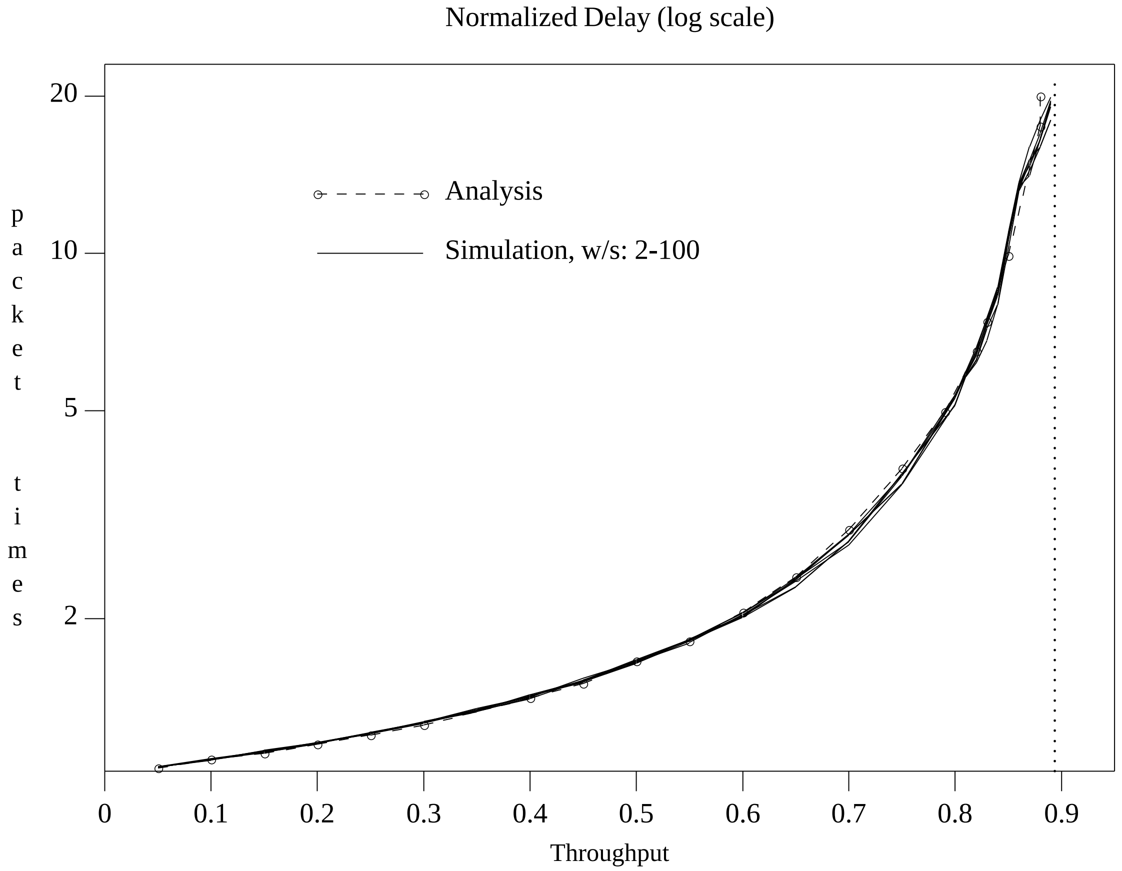


Figure 5.4: Normalized Delay of VT-CSMA/CD (model validation)

The results are shown in Figure 5.4 which illustrates the average message delay normalized to the message transmission time. The analytical results have been taken directly from the above paper. It can be seen the remarkable agreement between the outputs of the simulation and the analytical model, which displays the validity of our simulation model. We observe that all delay curves (i.e., each corresponds to an open system composed of a different number of workstations) are very close together which proves the predictability of the protocol behavior. According to the simulation statistics no packet was dropped and the average number of retransmissions was 0.7 in the worse case. This reveals the minimal contribution of the backoff algorithm to the average delay, which results in this remarkable agreement between simulation and analysis.

Besides the validation experiment, we carried out another test in order to compare the performance of Ethernet and VT-CSMA/CD in the context of the open systems. Thus, we consider the system described in section 4.2 which reflects a worst-case star Ethernet network according to the Ethernet specifications. All system and input traffic parameters

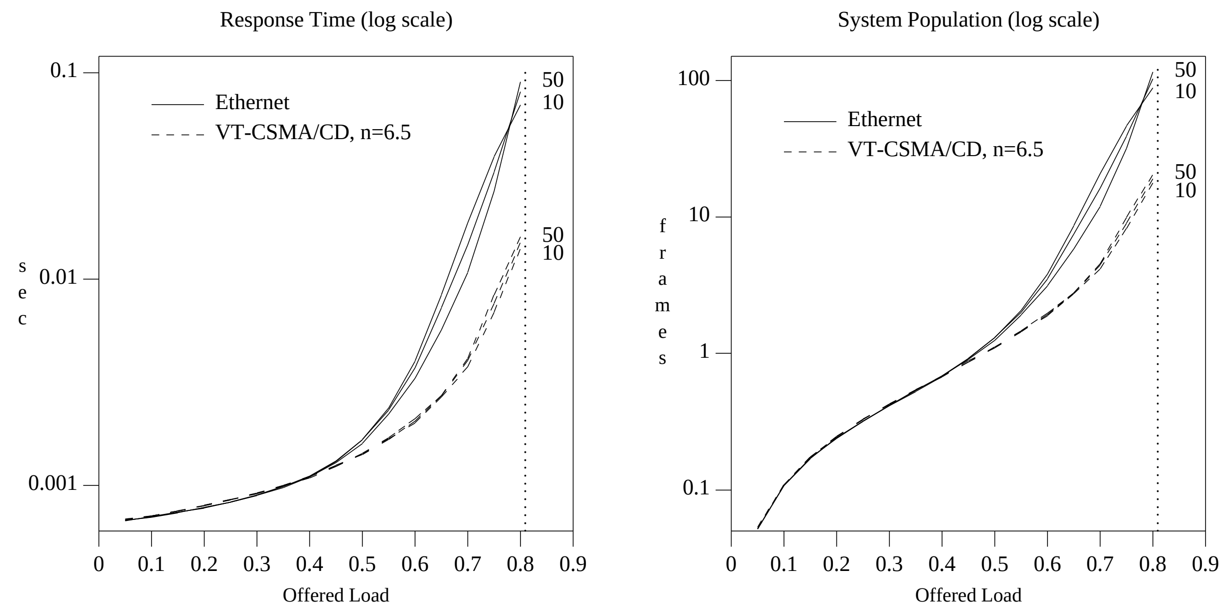


Figure 5.5: VT-CSMA/CD Delay and Population in an open system

remain the same. According to these parameters a and c are defined to be 0.0366 and 0.0076 respectively. Calculations based on the formulas described in the previous section yield $\eta^* = 6.61$ and capacity equal to 0.81. In our simulation we have used $\eta = 6.5$ and we have run the cases of $m = 10, 20$ and 50 workstations. The results for both protocols are contrasted in Figure 5.5. Curves in this figure have been labelled by the number of stations contained in each particular system. The vertical dotted lines denote the theoretical capacity of the network according to the VT-CSMA protocol.

We can see the enhanced performance of the VT-CSMA/CD protocol for loads over 0.45. Prior to this point both protocols have almost identical delay characteristics. VT-CSMA/CD delay curves does not show any overcrossings. The percentage of dropped packets is almost zero for all cases. The average number of retransmissions is bounded by 1.7, which shows the minimal contribution of the backoff algorithm. We also observe that the relative difference in the delay curves between the three different systems (according to the value of m) is negligible.

5.4 VT-CSMA/CD in Closed Systems

In this section we consider the VT-CSMA/CD protocol in the closed system architecture that we described in section 4.3. We substituted the Ethernet subsystem with the VT-CSMA/CD access control scheme in order to compare the performance of the two protocols,

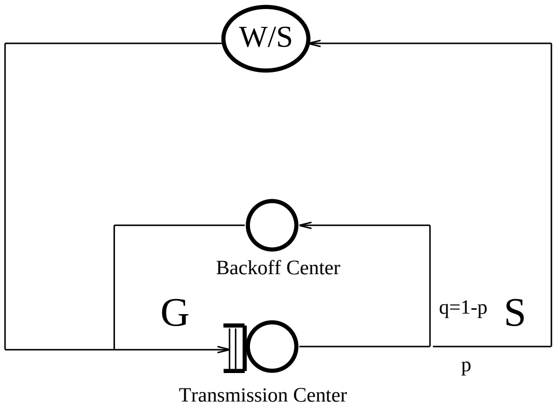


Figure 5.6: MVA model for the VT-CSMA/CD subsystem

in this type of environment. We also developed a very accurate analytic model for this closed system based on MVA-techniques. We tested cases where no processing (i.e., $D = 0$) or a little processing (i.e., $D = 5$ msec) delay at the workstations is involved. When the number of frames in the system is large (i.e., $n > 200$) we applied a modification of the VT-CSMA/CD mechanism for dealing with bulk arrivals, which enabled us to overcome system initialization problems. The results of all these experiments are presented here.

5.4.1 Analytical modeling issues

An MVA-based model for the VT-CSMA/CD subsystem in the closed system environment is shown in Figure 5.6. This model consists of a queuing center (termed "transmission" center) and a delay center (termed "backoff" center). The latter corresponds to packets that fail to be successfully transmitted and thus have to be rescheduled according to the binary exponential backoff algorithm. This justifies the use of the notation G and S for the traffic that flows in the transmission center (i.e., total traffic) and the traffic that flows out of the subsystem (i.e., throughput) respectively. p denotes the probability of success per attempt and q the probability that retransmission is needed. It is easy to see that $p = S/G$ and $q = 1 - p = (G - S)/G$. The visit counts at the two centers can be derived from G, S, p and q in a fashion similar to the one described in section 4.3.3. Thus, we can see that $V_t = G/S$ and $V_b = V_t - 1 = (G - S)/S$.

The first approach to the problem is to neglect the backoff center because of the fact that retransmissions are rare, and therefore they do not affect the system performance significantly. In this case we need to define the service demand at the transmission center only. As we already know this demand is the product of the visit count, V_t, and the service time per visit to the center S_t. In order to estimate S_t we use the notion of "work" that a single customer brings to the system upon its arrival. This work, X, is defined in [5] to consist of a *resting* component X_R (during which the virtual clocks are not running because the channel is sensed busy), and a *scanning* component X_S (during which the virtual clocks advance at a rate η trying to clear the backlog). It is obvious that the resting component is equal to the packet transmission time. The scanning component can be estimated from the fact that the virtual clocks must regain the "lost" (due to X_R) time during the scan time. When the virtual clocks run at rate η the backlog decreases at rate $\eta - 1$ (since real clock advances at rate unity). Thus $X_S = X_R/(\eta - 1)$ and $X = X_R \cdot \beta$, where $\beta = \eta/(\eta - 1)$. We are using the above formula to define the service time at the backoff center as

$$S_t = (P/C) * \beta \tag{5.7}$$

The solution to the model for the network subsystem is based on iterative solutions of the above MVA model, until successive values of a performance measure (e.g. throughput) are sufficiently close. At each iteration we update the visit count V_t. This is done by providing a better estimate for the total traffic G based on "inverting" the function $S = \mathcal{U}(G)$, where $\mathcal{U}$ is given by 5.4. Each time we solve the above MVA model we obtain the throughput S, and use it to determine the value of G that corresponds to the particular value of S, and thus redefine the visit count $V_t = G/S$. This iterative process is summarized below:

```
V_t = 1;
Solve MVA model;
if n = 1 then return
else if S > capacity then
      S = capacity;
repeat
      G = U^{-1}(S);
      V_t = G/S;
      Solve MVA model;
until successive values of S are sufficiently close
```

The former of the two special cases that we pointed out in the above description refers to a system with population of 1 customer. In this case we know that no retransmissions will occur so $G = S$. The latter special case applies when the initial estimate of the throughput, S, exceeds the maximum attainable throughput yielded by the analytic model (capacity). If this happens we cannot "invert" the function $\mathcal{U}$. Therefore we set the first estimate to the value of the capacity before we proceed. The fact that in VT-CSMA/CD the dominant part of the message delay is not attributed to the retransmissions guarantees that the executions of the main loop of the above solution will be very few before the model converges.

In order to increase the accuracy of the model even more, we can estimate the average backoff delay and set the service demand at the backoff center. The demand at the backoff center is the product of the visit count V_b and the service time per visit to this center S_b. We know that p denotes the probability that the current retry is enough. We need to find the proportion of packets that needed a 1st, 2nd, 3rd, 4th, ..., and a 15th backoff. Assuming the probability of success is the same on every attempt, these proportions are 1, $(1-p)$, $(1-p)^2$, $(1-p)^3$, ..., and $(1-p)^{14}$ respectively. The mean backoff delay (in slot times) for the packets that need i backoffs is $2^i/2 = 2^{i-1}$ if $1 \le i \le 10$ or $2^{10}/2 = 2^9$ if $11 \le i \le 15$. Thus, the mean backoff delay is given by the weighted sum of these backoff delays normalized by the sum of the proportions, yielding:

$$
\begin{aligned}
S_b &= \frac{\sum_{i=1}^{10} 2^{i-1} \cdot (1-p)^{i-1} + \sum_{i=11}^{15} 2^9 \cdot (1-p)^{i-1}}{\sum_{i=1}^{15}(1-p)^{i-1}} \\
&= \frac{1 - 512 \cdot (1-p)^{10} \cdot [1-(1-p)^5]}{1-(1-p)^{15}}
\end{aligned} \tag{5.8}
$$

Now we can set the service demand at the backoff center using the above formula, in order to get more accurate results for the system performance. As we will see in the following subsections the two models (i.e. with or without the backoff delay center) yield very similar results. This allows us to adopt the non-backoff delay version for getting acceptable predictions for the network performance, without high level of computation.

5.4.2 The D=0 case

The simulation results for the delay and throughput of the network along with the predictions of the two versions of the analytic model, that we described previously, are shown in Figure 5.7. Curves have been labelled by the number of the stations in the system. Model 1 assumes no delay at the backoff center, whereas model 2 refers to the (more complex) case of an extra delay at this center. We can see that all configurations perform very similarly and that there are practically no overcrossings in the throughput and delay curves. Both analytic models give very accurate predictions. These observations underline the advantage of the VT-CSMA/CD protocol over Ethernet, which means that the former protocol has an accurate analytic representation unlike Ethernet whose performance was found to depend on the details of the system (numbers of stations, traffic patterns, etc.). The accuracy of the analytic model can be attributed to the minimal contribution of the backoff algorithm which has proven hard to model.

Figures 5.8 and 5.9 show a comparison of the two protocols in this closed system context. VT-CSMA/CD is characterized by substantially better average throughput and delay and much lower sensitivity to the system configuration. It is easy to see that the network throughput remains almost constant independent of the number of frames in the system. In other words the network utilization remains the same, which demonstrates stability. On the other hand Ethernet throughput decreases dramatically in the first region of $n \in [2, 30]$ frames. The variance of the network delay for VT-CSMA/CD is minimal in comparison

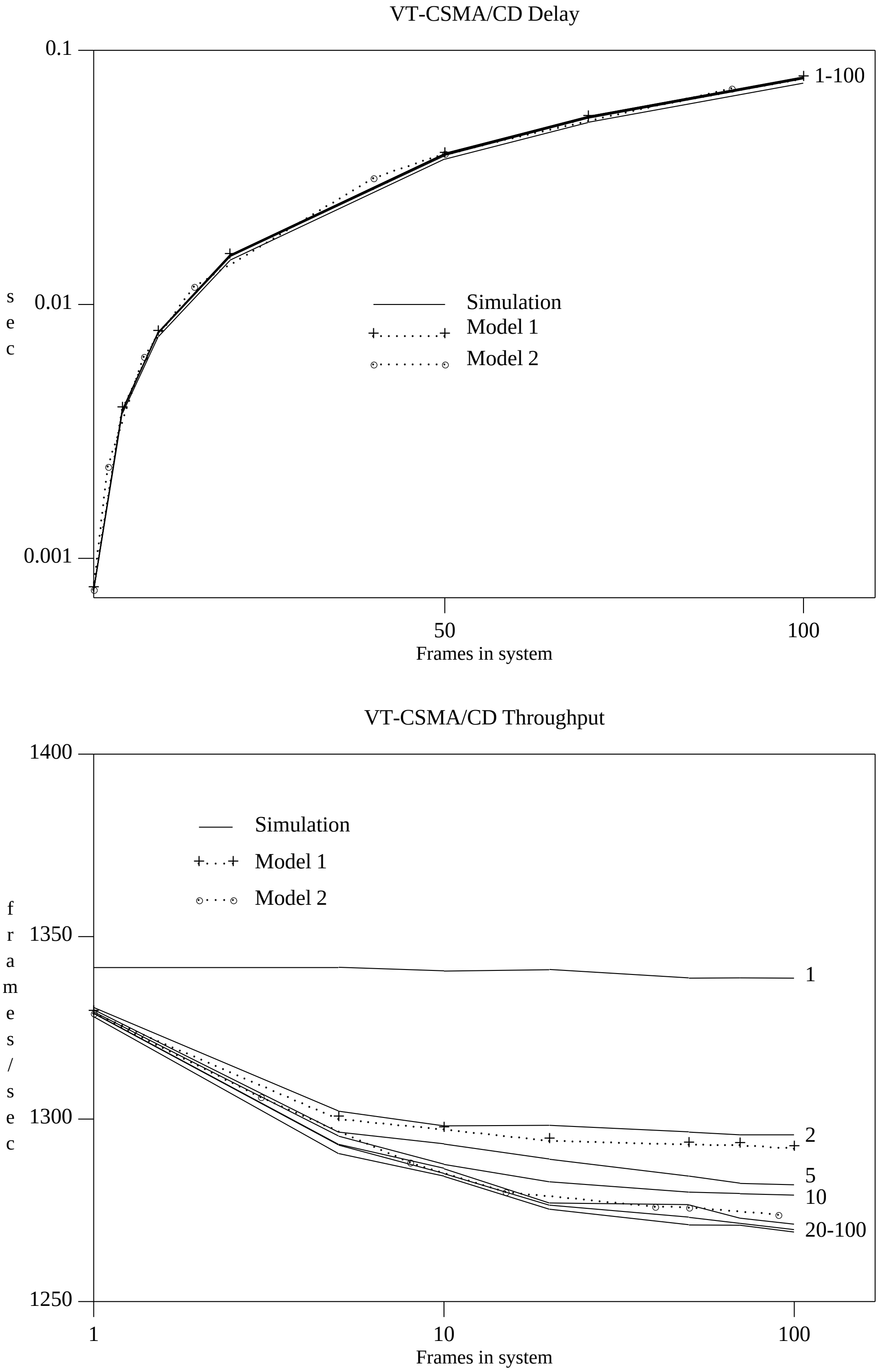
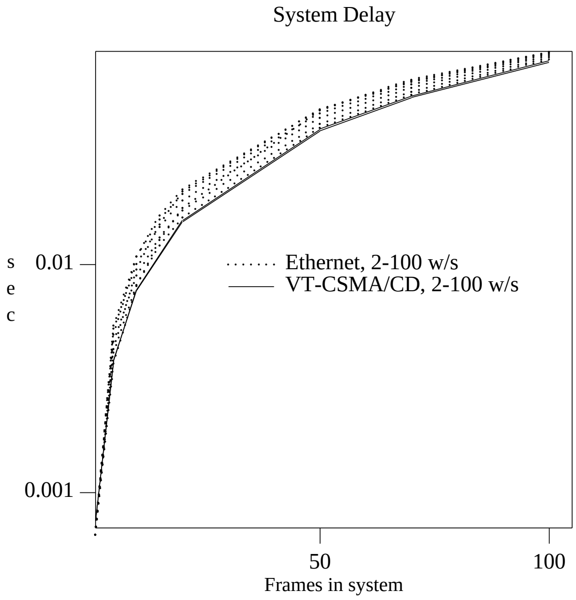


Figure 5.7: VT-CSMA/CD Performance in a closed system with D=0



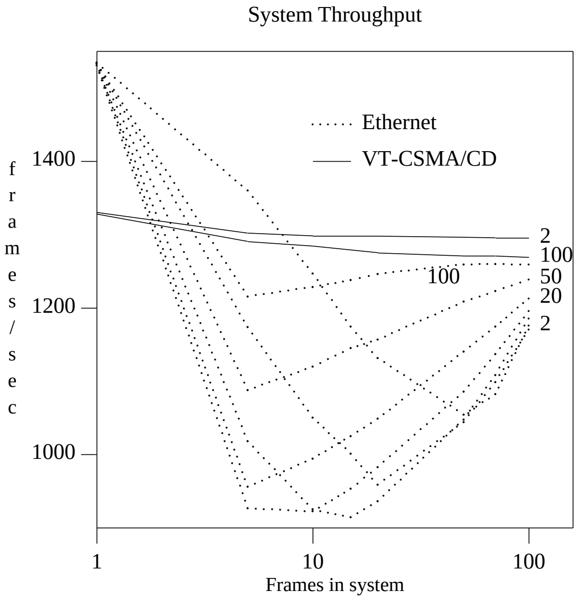


Figure 5.8: VT-CSMA/CD Performance vs. Ethernet Performance, D=0 (a)

to the Ethernet, which ranges up to a third of the value of the average delay, thus making it unsuitable for time critical applications. The percentage of errors is essentially zero for VT-CSMA/CD. VT-CSMA/CD does not monopolize the channel as it makes very little use of the retransmission algorithm. This is shown by the average number of retransmissions per frame, which does not exceed 1.4.

The value of η that we used in this simulation run is based on η^*, which is approximately equal to 6.5 as in the case of the open system. The channel utilization remains closed to 81% cases. The analytic models converge in at most 3 iterations.

5.4.3 The D=5 msec case

The results for this more realistic case are shown in Figure 5.10. The analytic model[4] appears to be very accurate, for the cases when the network is the bottleneck center (i.e., $m \geq 10$) as well as when the workstations are saturated (i.e., $m < 10$).

Figure 5.11 contrasts these results to the ones obtained from the Ethernet-based system, for systems where the m is chosen so as the network is the performance bottleneck of the system (i.e., $m \geq 10$). We can see that in all cases VT-CSMA/CD outperforms Ethernet in the primary performance measures as well as the variance of the delay and the percentage of dropped packets. We also observe that in the region where the network is saturated, the throughput curves are much closer than in the $D = 0$ case, which shows that the protocol

[4]We have applied only Model 1, which has no backoff delay center, for simplicity in the calculations.

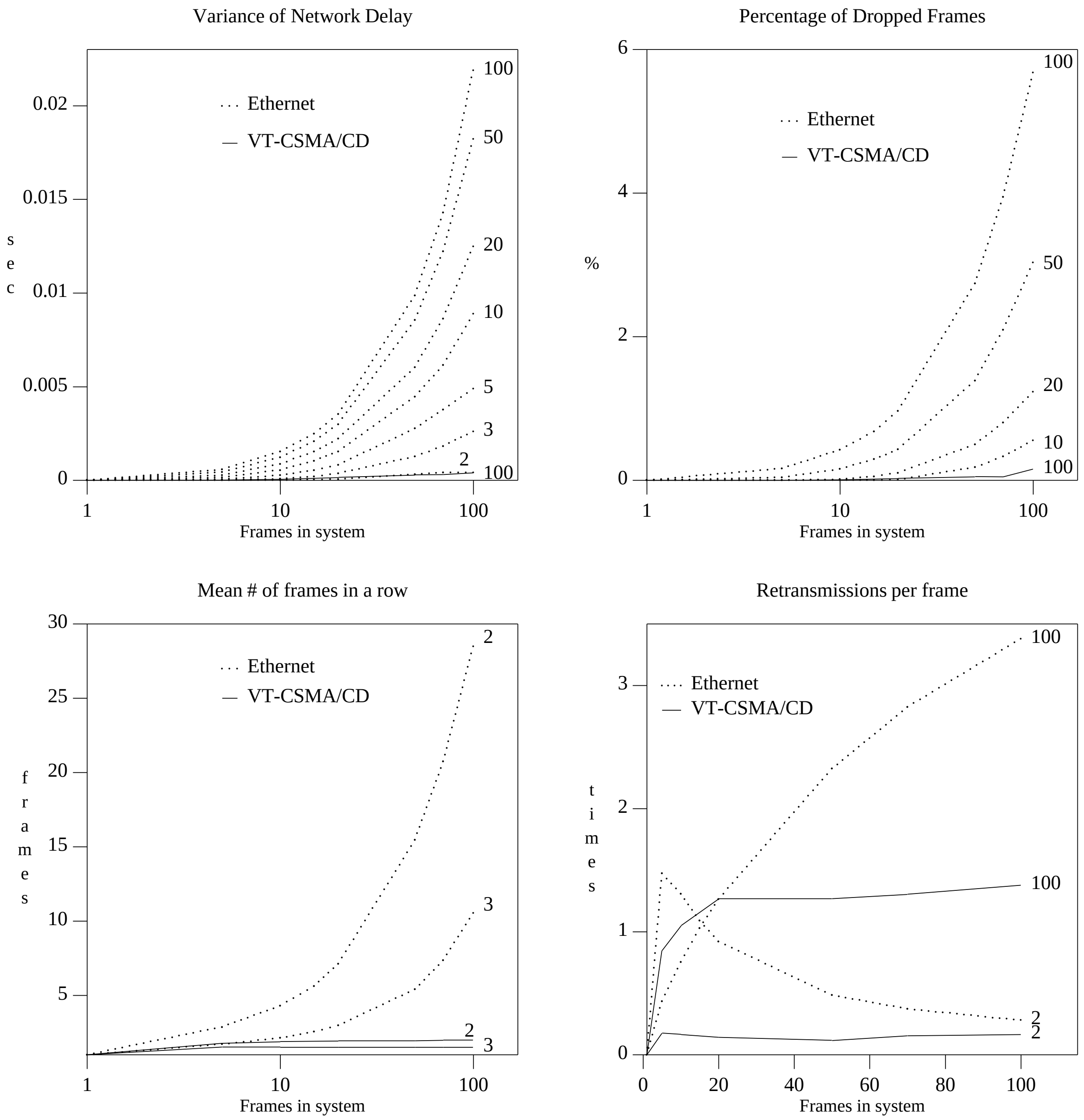


Figure 5.9: VT-CSMA/CD Performance vs. Ethernet Performance, D=0 (b)

behavior is more predictable in this case. The value of η that we used that we used here is the same as the one in the previous (i.e., $D = 0$) case.

5.4.4 A modification of VT-CSMA/CD

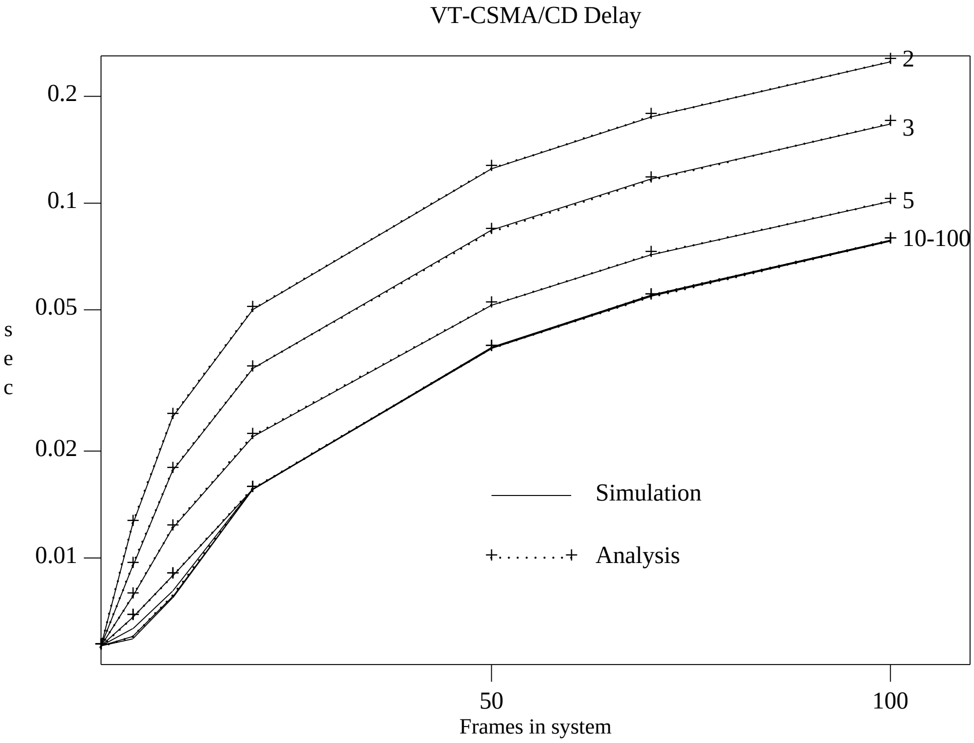
Finally, we will illustrate a case that caused several problems in the context of the closed system that we examined before. When the number of frames in the system was increased beyond 200, we could not initialize the system properly, due to massive collisions at the beginning of the simulation execution. These collisions increased the value of G to a level beyond the network capacity. Since the initialization of the system is done by means of an external Poisson arrival process, we could solve this problem by gradually decreasing the intensity of the arrival process. However, it is not trivial to determine the appropriate way to update the intensity parameter of the process. Thus since similar problems could occur in other contexts (e.g. when there are lots of bulk packet arrivals, such as when the traffic is dominated by large messages, which must be fragmented into multiple packets), we applied a modification to the protocol operation which alleviated the problem.

Each time a frame experiences a collision (and a random backoff delay is determined by the binary exponential backoff algorithm), we now assume that the virtual clock of each colliding workstation stops running until the backoff delay has elapsed. This scheme prevents subsequent frames in a burst arrival to jump in front of the colliding frame in the transmit queue of the station thus causing more collisions. It also eliminates the need for sorting the transmit queue. However, it achieves higher throughput than the theoretical capacity if the value of η is set to a large value, and therefore cannot be predicted by the analytic model that we used in the previous two sections.

The performance of this modified scheme is compared to the performance of the Ethernet in Figure 5.12. The value of η is set to 20. We can see that the modified VT-CSMA/CD performs better for systems with large numbers of workstations. For smaller systems Ethernet achieves higher throughput mainly because of the monopolization phenomenon, which increases both the variance of the delay and the unfairness of the network. The modified scheme keeps the percentage of dropped packets to zero and relies very little on the backoff algorithm, since the average number of retransmissions per frame is bounded by 1.8 in the worst case.

5.5 "1 persistent" behavior of VT-CSMA/CD

In this section we will present the results of an experiment conducted on the closed system where Boggs et al. carried out their measurement study. The Ethernet subsystem has been replaced by the VT-CSMA/CD protocol. This system is described in section 4.1. Assuming packets of constant size (namely $P = 64$ bytes), the parameters a and c have



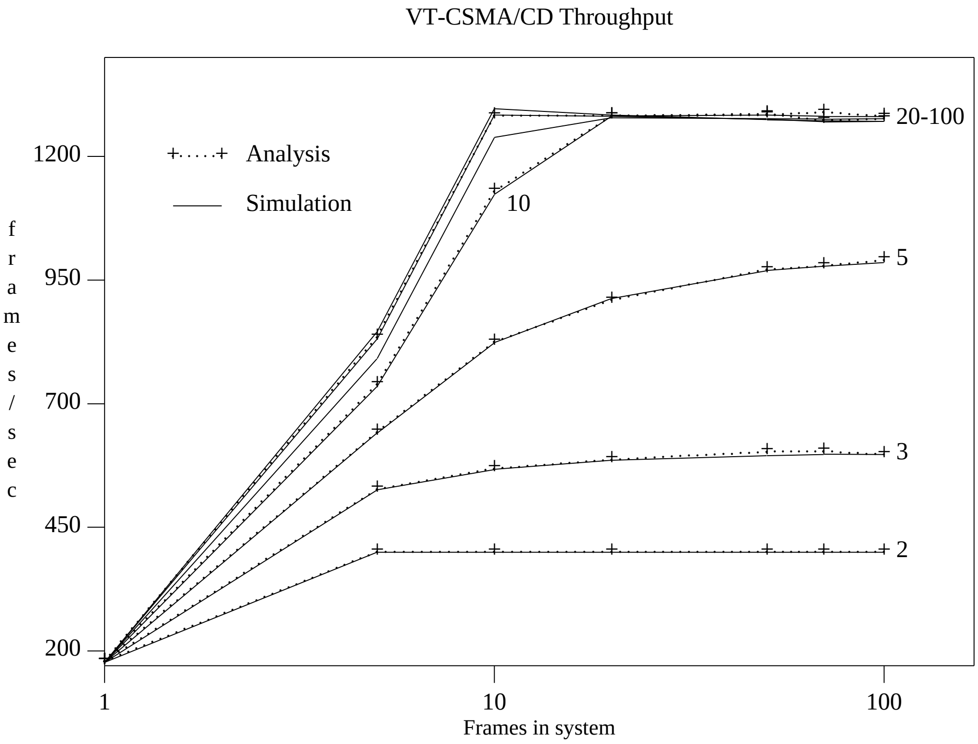


Figure 5.10: VT-CSMA/CD Performance in a closed system with D = 5 msec

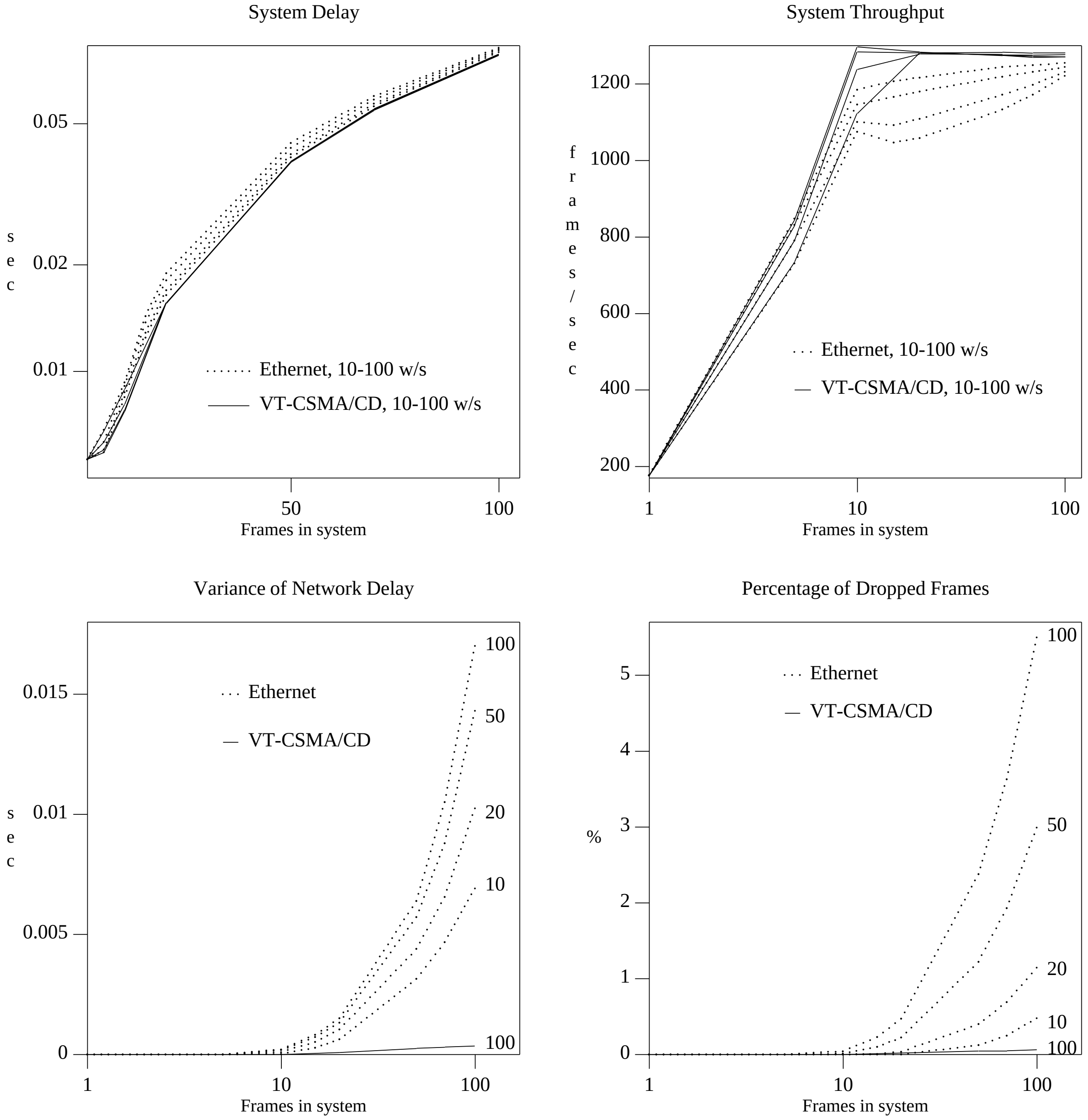


Figure 5.11: VT-CSMA/CD Performance vs. Ethernet Performance, D = 5 msec

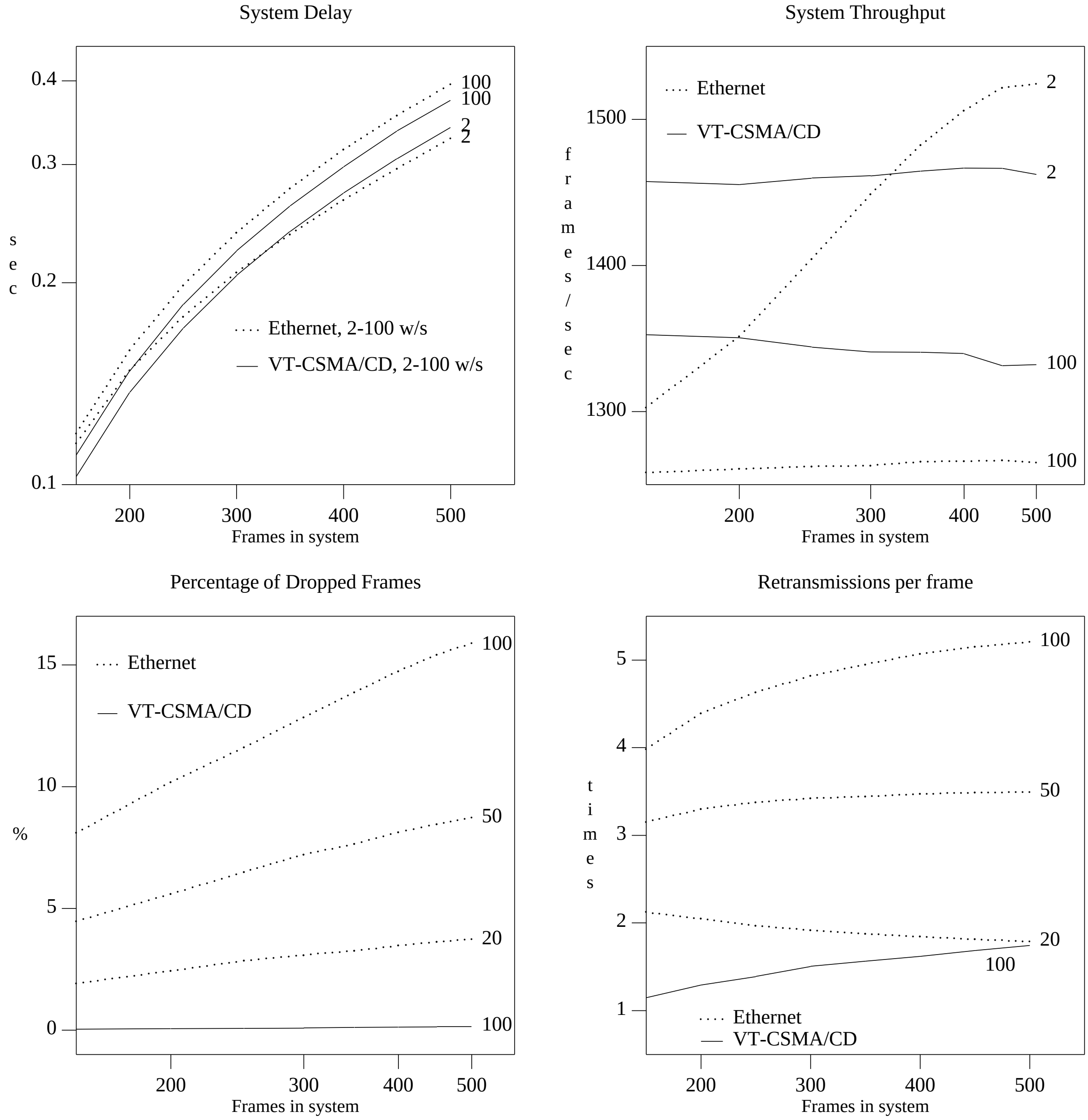


Figure 5.12: Modified VT-CSMA/CD vs. Ethernet

been set to 0.031 and 0.0682 respectively. For these values the analysis yields optimum virtual clock rate $\eta \simeq 7.5$ and capacity 0.82.

We have tried two instances of the VT-CSMA/CD protocol. In the first one the clock rate is set to 7 whereas in the second one it is set to 50. The results are illustrated in Figure 5.13 together with the Ethernet performance. We can see that when $\eta = 7$, VT-CSMA/CD outperforms Ethernet in most cases except for systems with few workstations (i.e., when m is less than 10). The standard deviation of the response time and the throughput, as well as the percentage of dropped packets and the collision rates are close to zero.

However, as the virtual clock rate increases the performance of VT-CSMA/CD resembles the performance of the Ethernet. This can be seen clearly for the case when $\eta = 50$. This verifies the analytic result which states that 1-persistent CSMA is a special case of VT-CSMA when the virtual clock rate approaches infinity.

We should point out that in this system each station holds a single frame which is destined to itself. This results in a synchronization of the stations in the system yielding minimal collisions and very high bit rates. Indeed, the channel utilization has been measured above 90%, whereas the theoretical maximum is closer to 82%. This synchronization is caused by the fact that when a station completes a successful packet transmission at some time t, the next arrival time for the packet will be at t plus the (constant) processing time. Since no other station had a packet departure at t, none will have an arrival at the same time as the successful packet. Thus, its next transmission it is almost guaranteed not to interfere with any other frames from another station because of the operation of the virtual clocks.

5.6 VT-CSMA in a Client/Server System

In the last section of this chapter, we illustrate the performance of the VT-CSMA/CD protocol in the client/server architecture that we described in section 4.4. We assume constant-size packets of 64 bytes and 5 frames per station. For these parameters, the virtual clock rate is set to $\eta = 7$.

Figure 5.14 presents the results and compares them to the Ethernet ones. It is obvious that VT-CSMA/CD has better delay and throughput characteristics, it keeps the percentage of dropped packets close to zero and the number of retransmissions per frame close to 1. We can also see that VT-CSMA/CD has minimal effect to the system response time before it saturates, which is desirable in real systems. On the other hand, Ethernet gradually starts making contributions to the system delay before it saturates. These characteristics prove the superiority of VT-CSMA/CD over Ethernet in this client/server distributed computing environment.

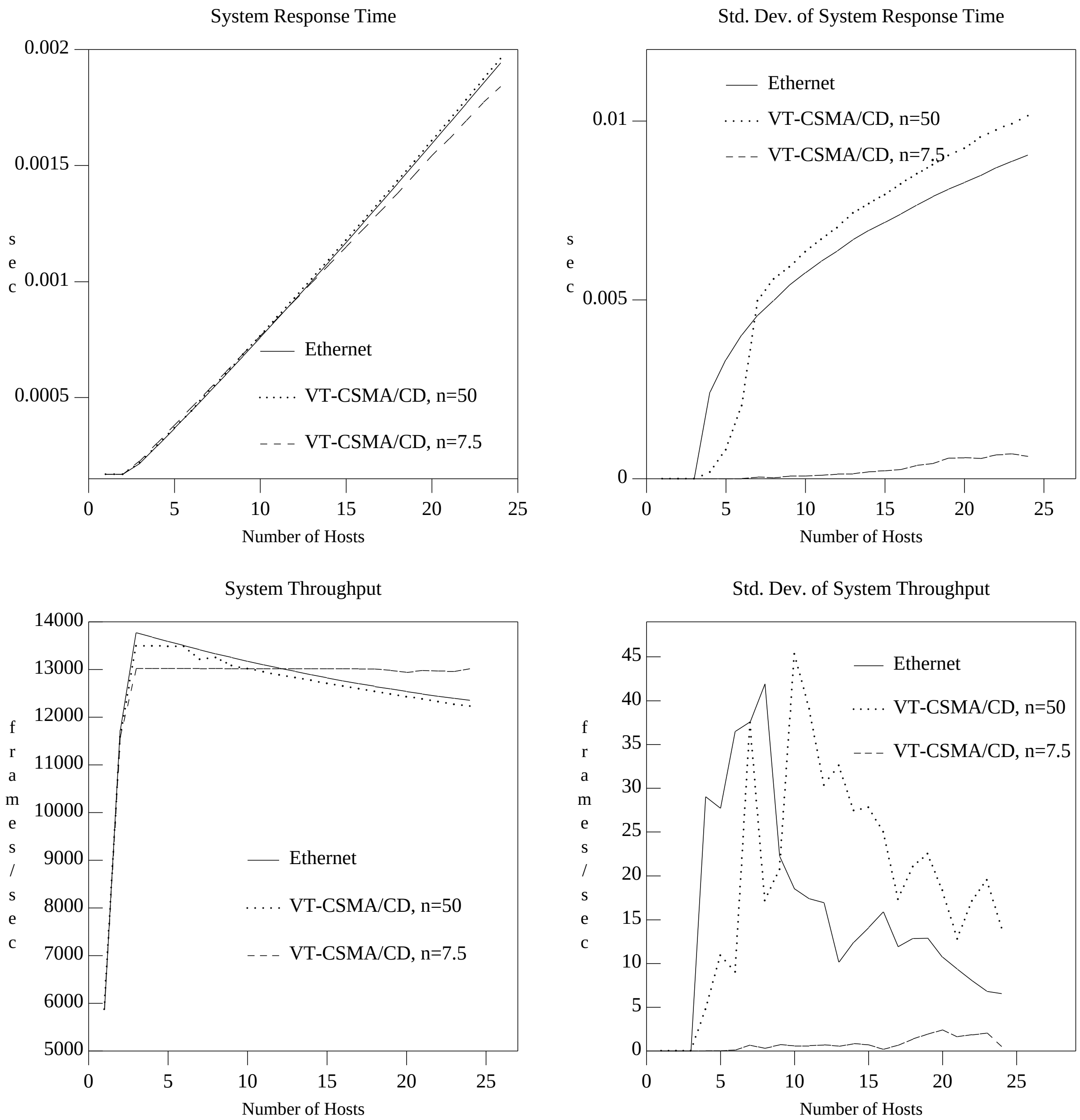


Figure 5.13: "1-persistent" behavior of VT-CSMA/CD

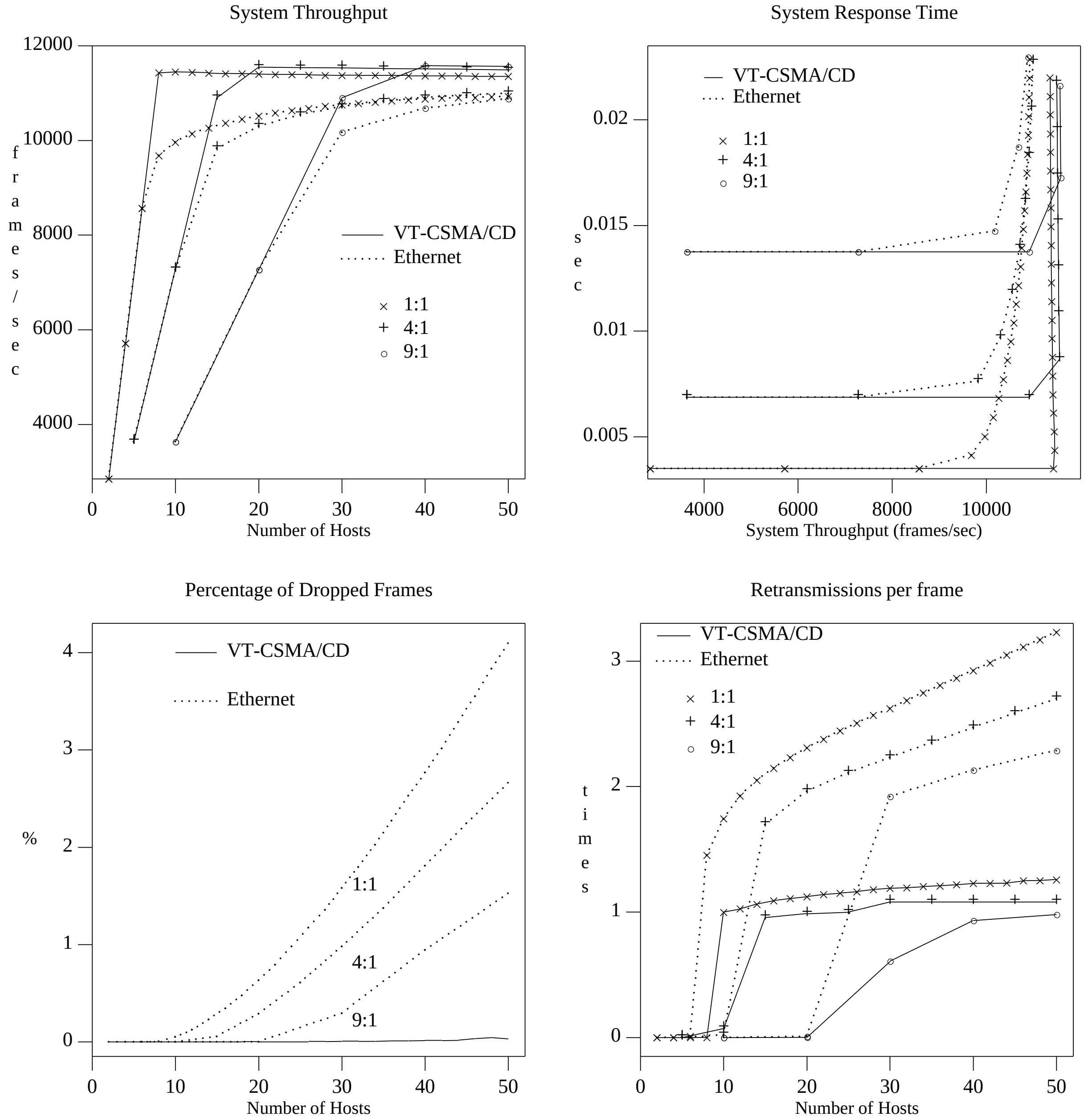


Figure 5.14: Ethernet vs. VT-CSMA/CD in a client/server architecture

Chapter 6

Epilogue

Ethernet networks have become very popular during the past decade, because of their ability to support a wide variety of distributed computing environments successfully. Measurements studies suggest that Ethernet is capable of good performance for typical applications. Analysis ensures this claim for cases of low to moderate loads. However, today's state-of-the-art workstations in conjunction with efficient communication protocols, are capable of generating workloads that Ethernet of the 80's did not experience. Analytical studies have not been able to model the function of Ethernet's truncated binary exponential backoff algorithm, which determines the protocol behavior in a high network demand regime. Therefore, the only feasible way to study the characteristics on a loaded Ethernet in an accurate fashion is via simulation.

For the purposes of this report, we have constructed a detailed simulation model that accurately emulates the Ethernet protocol on the equidistant-star topology. The novelty of this model is that it has the additional ability of representing product-form queuing networks that include the network as a service center. In order to verify the accuracy of our model, we set up the simulator to represent a real system for which sufficient detail is available in the literature, and compared our simulation output to the published results of the measurement study on this particular system. After showing that we could reproduce these results with reasonably good agreement (validating our model and displaying the expressive power of simulation techniques to accurately reflect the behavior of real systems), we extrapolated this model to a number of similar configurations. Our results show that existing analytic studies are unable to capture the protocol behavior under high-loads in both *open* and *closed* systems.

In the case of open systems, the agreement between simulation and analysis stops when the offered load exceeds 50% at which point the details of the Ethernet protocol (its backoff algorithm in particular) start to matter. Thereafter, the Ethernet performance depends on the number of "nodes" that are attached to it. Surprisingly, systems with fewer workstations appear to have worse performance under heavy load and are less fair than larger systems. The unpredictable behavior of Ethernet is almost exclusively attributed to the behavior of its backoff algorithm.

We have also cleared out a misinterpretation of the term *inter-arrival times* as it is used in the context of measurement studies. We have pointed out that the appropriate term for what network practitioners call inter-arrival times is *inter-departure* times, and we have pointed out that the claim that non-exponential departures imply non-exponential arrivals is not logical. We then continued by measuring the inter-departure times (in our simulator) corresponding to exponential arrivals, and found that the combined effects of the access protocol and the backoff algorithm introduce distortion in the arrival process, thus making the departure process non-exponential regardless of the input process.

In the case of closed systems, we consider a fixed number of circulating customers and little processing between successive network requests. The existing analytic model appears to be accurate for systems where the choice of parameters does not result in network saturation (i.e., the workstations are the "bottleneck" devices). However, as the number of workstations increases, the network starts making contributions to the average system response time, and finally saturates. Configurations that saturate the Ethernet can have dramatically different performance according to the number of workstations that they include. The analytic model cannot distinguish between them and is incapable of reflecting their behavior.

As the processing time between successive requests decreases, the Ethernet behavior becomes even more unpredictable. An initial sharp decrease in the throughput in the region corresponding to low population levels, reveals severe performance degradation. Systems composed of small numbers of workstations appear to be more unpredictable in terms of the analytic model predictions, due to the intrinsic unfairness of Ethernet for small configurations. Longer queuing delays, increase the variance of the network delay, as frames in the transmit queues of the stations pile up behind frames that experience long backoff delays. Under high-load circumstances, a channel monopolization phenomenon is in effect. Stations tend to send all packets in their transmit queues when they capture the channel, thus making the network display a last-come-first-served policy.

We have investigated the possibility of developing a better analytic model by incorporating a second service center to represent the backoff algorithm and hence provide a better estimation of the instantaneous load of the network. Unfortunately, instrumenting our simulation to give the necessary parameters for such a model shows that these parameters are very sensitive to external factors, such as packet sizes, traffic patterns and number of stations. Thus it seems unlikely that a tractable generally usable analytical model of an Ethernet subsystem that is better than the Almes and Lazowska model will be found.

It is also shown that in cases of heavy network load, we can enhance the Ethernet performance by imposing a gap of length slightly larger than the round-trip propagation delay of the network, between successive packet transmissions from every station. Finally, integrating our model into a simple client/server architecture, we showed that for a user-interaction intensive workload (where the network becomes a performance bottleneck) the details of the Ethernet protocol affect the calculation of the system response time and should not be ignored. The network contribution to the delay is even more evident when we divide stations into pairs, having the two stations of each pair interchange messages between one another.

The profound effect of the backoff algorithm on the Ethernet performance, motivated the investigation of the predictability of the behavior of other CSMA protocols that do not rely so heavily on the retransmission strategy, and the comparison of their performance to the Ethernet. Virtual Time CSMA/CD (VT-CSMA/CD) is a media access algorithm of this sort. In order to make the above comparison feasible, we extended the Ethernet simulation model to represent the VT-CSMA/CD operation. The extended model has been validated against the accurate analytic model of the VT-CSMA/CD protocol. For the case of the closed system we have developed an accurate analytic model based on iterative solutions of an MVA model. In both cases of open and closed systems, VT-CSMA/CD behavior is according to the analytic model predictions. Moreover, its sophisticated transmission scheduling mechanism, achieves better performance than the Ethernet, largely by avoiding the unpredictable effects of the binary exponential backoff algorithm.

An obvious future direction of this work would be the development of an accurate analytic model of the Ethernet backoff algorithm. Its applicability would be immediate, as recent technological advances impose gradually increasing levels of network demand. As far the simulation model is concerned, an extension to a more detailed client/server computing environment, based on a real system and network description, may clearly prove the impact of network congestion to the system performance.

Unlike the sentiment expressed by Boggs et al. [29], we believe that simulation is indeed capable of extending the results of measurement studies beyond the limitations of the available system resources, with impressive accuracy. Future measurement studies should provide sufficient information about the network operational characteristics to enable this extension.

Appendix

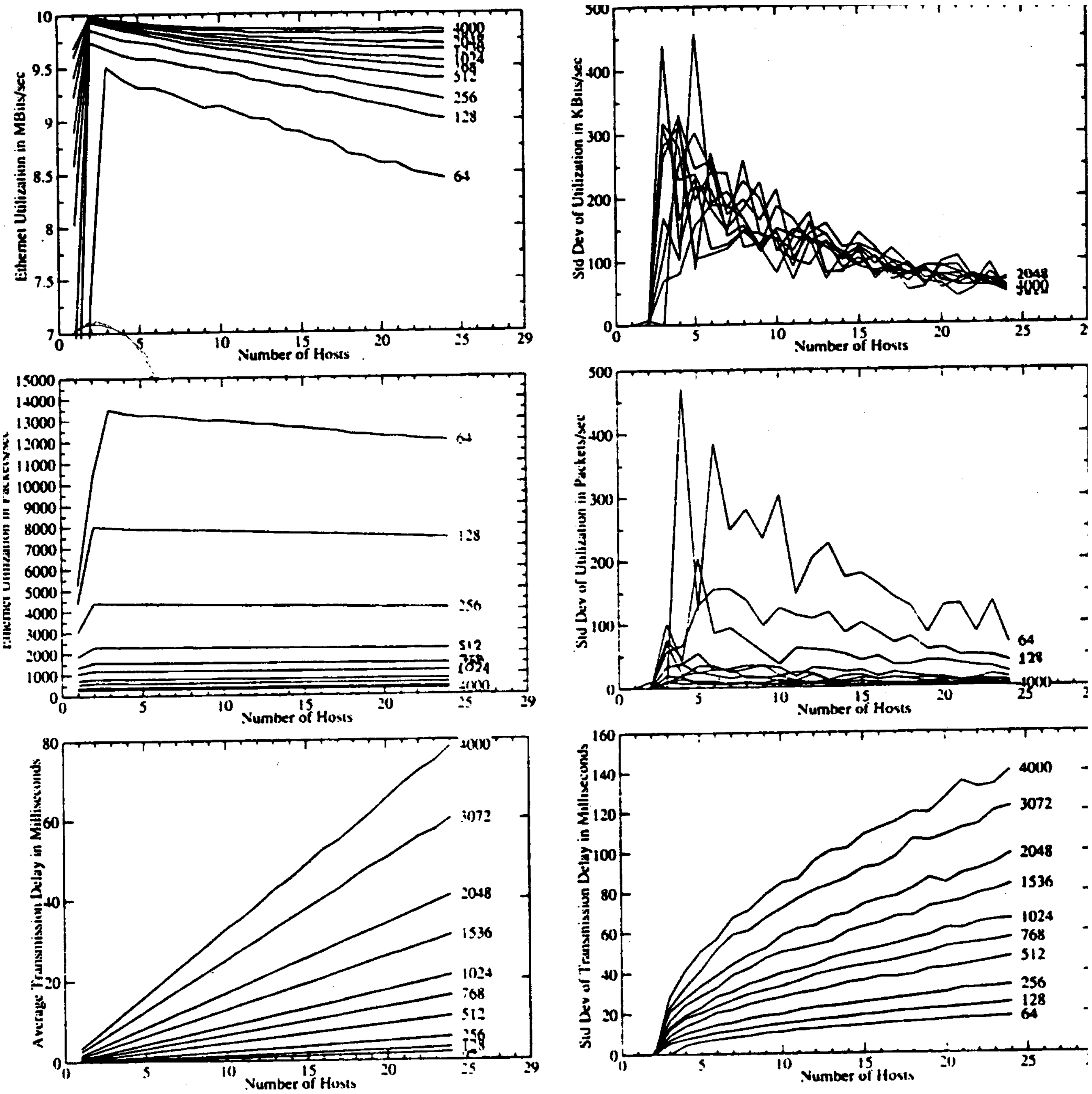


Figure A.1: Measured performance of Ethernet (reprint from Boggs et al.)

Bibliography

[1] R. Binder, N. Abramson, F. F. Kuo, A. Okinaka, and D. Wax. ALOHA Packet Broadcasting – A Retrospective. In *AFIPS Conference Proceedings, NCC 44*, pages 203–251, 1975.

[2] Leonard Kleinrock and Fouad A. Tobagi. Packet Switching in Radio Channels: Part I – Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics. *IEEE Transactions on Communications*, 23(12):1400–1416, December 1975.

[3] Robert M. Metcalfe and David R. Boggs. Ethernet:Distributed Packet Switching for Local Computer Networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[4] Mart L. Molle. Unifications and Extensions of the Multiple Access Communications Problem. Technical Report CSD Report No. 810730 (UCLA-ENG-8118), Computer Science Department, University of California, Los Angeles, July 1981.

[5] Mart L. Molle and Leonard Kleinrock. Virtual Time CSMA: Why Two Clocks Are Better than One. *IEEE Transactions on Communications*, 33(9):919–933, September 1985.

[6] Mart L. Molle. Prioritized-Virtual Time CSMA: Head-of-the-Line Priority Classes without Added Overhead. To appear in *IEEE Transactions on Communications*, 1991.

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

[8] Guy T. Almes and Edward D. Lazowska. The Behavior of Ethernet-like Computer Communications Networks. In *Proceedings of the Local Area Communications Network Symposium*, Boston, MA, May 1979.

[9] S.S. Lam. A Carrier Sense Multiple Access Protocol for Local Networks. *Computer Networks*, 4(1):21–32, February 1980.

[10] Mischa Schwartz. *Telecommunication Networks: Protocols, Modeling and Analysis.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1987.

[11] Jeremiah F. Hayes. *Modeling and Analysis of Computer Communication Networks.* Plenum Press, New York, 1984.

[12] F.A. Tobagi and V.B. Hunt. Performance Analysis of Carrier Sense Multiple Access with Collission Detection. *Computer Networks*, 4(5):245–259, October 1980.

[13] Nachum Shacham and V. Bruce Hunt. Performance Evaluation of the CSMA/CD (1-persistent) Channel-Access Protocol in Common-Channel Local Networks. In *Proceedings of the IFIP TC 6 International In-Depth Symposium on Local Computer Networks*, Florence, Italy, April 1982.

[14] Edward J. Coyle and Bede Liu. Finite Population CSMA/CD Networks. *IEEE Transactions on Communications*, 31(11):1247–1253, November 1983.

[15] Edward J. Coyle and Bede Liu. A Matrix Representation of CSMA/CD Networks. *IEEE Transactions on Communications*, 33(1):53–64, January 1985.

[16] K. Sohraby, M.L. Molle, and A.N. Venetsanopoulos. Comments on "Throughput Analysis for Persistent CSMA Systems". *IEEE Transactions on Communications*, 35(2):240–243, February 1987.

[17] Hideaki Takagi and Leonard Kleinrock. Throughput Analysis for Persistent CSMA Systems. *IEEE Transactions on Communications*, 33(7):627–638, July 1985.

[18] Timothy A. Gonsalves and Fouad A. Tobagi. On the Performance Effects of Station Locations and Access Protocol Parameters in Ethernet Networks. *IEEE Transactions on Communications*, 36(4):441–449, April 1988.

[19] Shuji Tasaka. Dynamic Behavior of a CSMA-CD System with a Finite Population of Buffered Users. *IEEE Transactions on Communications*, 34(6):576–586, June 1986.

[20] Mart L. Molle. Analysis of a Class of Distributed Queues with Applications. In *Proceedings of the Seminar on Performance of Distributed and Parallel Systems*, Kyoto, Japan, December 1988.

[21] Dimitri Konstantas. Virtual Time CSMA: A Study. Master's thesis, Department of Computer Science, University of Toronto, 1983.

[22] M.L. Molle and D. Kostantas. A simulation Study of Retransmission Strategies for the Asynchronous Virtual Time CSMA Protocol. In *Proceedings Performance '83, 9th International Symposium of Computer Performance Modeling, Measurement, and Evaluation*, pages 295–308, May 1983.

[23] G.V. Guardalben. On the Coexistence of Different CSMA Protocols on the Same Network. Master's thesis, Department of Computer Science, University of Toronto, 1986.

[24] John F. Shoch and Jon A. Hupp. Measured Performance of an Ethernet Local Network. *Communications of the ACM*, 23(12):711–720, December 1980.

[25] Riccardo Gusella. The Analysis of Diskless Workstation Traffic on an Ethernet. Technical Report UCB/CSD 87/379, Computer Science Division, University of California Berkeley, December 1987.

[26] Riccardo Gusella. File Access and Paging Behavior of Diskless Workstations on an Ethernet. In *Proceedings of the International Seminar on Performance of Distributed and Parallel Systems*, Kyoto,Japan, 1988.

[27] Riccardo Gusella. Characterizing the Variability of Arrival Processes with Indices of Dispersion. To appear in *IEEE Transactions on Communications*, 1991.

[28] D.V. Wilson and W.E. Leland. High-Time Resolution Measurement and Analysis of LAN Traffic. In *Proceedings of the 1990 Bellcore Symposium on Performance Modeling*, Livingston, NJ, May 1990.

[29] David R. Boggs, Jeffrey C. Mogul, and Christopher A. Kent. Measured Capacity of an Ethernet: Myths and Reality. In *Proceedings of the SIGCOMM '88 Symposium on Communication Architectures and Protocols, ACM SIGCOMM*, Stanford, California, August 1988.
Reprinted as Technical Report by DEC Western Research Laboratory (WRL), Palo Alto, California, September 1988.

[30] Jeffrey P. Buzen. Computational Algorithms for Closed Queueing Networks with Exponential Servers. *Communications of the ACM*, 16(9):527–531, September 1973.

[31] Peter J. Denning and Jeffrey P. Buzen. The Operational Analysis of Queueing Network Models. *ACM Computing Surveys*, 10(3):225–261, September 1978.

[32] Terry James Coatta. Queueing Networks with State-Dependent Service Rates. Master's thesis, Department of Computer Science, University of Toronto, 1987.

[33] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.

[34] C. H. Sauer and K. M. Chandy. Approximate analysis of Central Server Models. IBM Journal of Reaserch and Development, May 1975.

[35] Michael K. Molloy. *Fundamentals of Performance Modeling.* Macmillan Publishing Company, New York, 1989.

[36] Leonard Kleinrock. *Queueing Systems, Volume I: Theory.* John Wiley & Sons, Inc., 1975.

[37] M. Reiser and S. S. Lavenberg. Mean-Value Analysis of Closed Multichain Queuing Networks. *Journal of the ACM*, 27(2):313–322, April 1980.

[38] Kenneth C. Sevcik and Israel Mitrani. The Distribution of Queueing Network States at Input and Output Instants. *Journal of the ACM*, 28(2):358–371, April 1981.

[39] Amdahl. MAP User's Guide, Version 2.0. Publication Number:MC-142131-004, February 1988.

[40] Amdahl. MAP Reference Manual, Version 2.0. Publication Number:MC-142130-004, February 1988.

[41] Edward D. Lazowska, John Zahorian, David R. Cheriton, and Willy Zwaenepoel. File Access Performance of Diskless Workstations. *ACM Transactions on Computer Systems*, 4(3):238–268, August 1986.

[42] Michael K. Molloy. Validation of MVA models for Client/Server Systems. Hewlett-Packard, 1990.

[43] Dinesh Mirchandani and Prabuddha Biswas. Characterizing and Modeling Ethernet Performance of Distributed DECwindows Applications. In *ACM Sigmetrics Conference*, Boulder, Colorado, May 22-25 1990.
Reprinted as Technical Report by DEC, Nashua NH, September 15, 1989.

[44] Digital, Intel, and XEROX. The Ethernet. A local Area Network. Data Link Layer and Physical Layer Specifications. Version 1.0, September 1980.

[45] Raymond L. Pickholtz, editor. *Local Area & Multiple Access Networks.* Advances in Telecommunication Networks. Computer Science Press, Rockville, Maryland, 1986.

[46] Andrew S. Tanenbaum. *Computer Networks.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1988.

[47] Renos Melas and Mart Molle. On the Ultimate Undesirability of "Blast" Protocols for Bulk Data Transfer over Ethernet. Technical Note-52, Computer Systems Research Institute, September 1989.

[48] Douglas Comer. *Internetworking with TCP/IP Principles, Protocols, and Architecture.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1988.

[49] James Martin. *Local Area Networks: Architectures and Implementations.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.

[50] Intel. Local Area Networking (LAN) Component User's Manual. Order Number: 230814-002, September 1985.

[51] Digital Equipment Corporation. DELQA User's Guide. Order Number: EK-DELQA-UG-001, August 1987.

[52] Gary Stix. Telephone wiring: a conduit for networking standards. *IEEE Spectrum*, pages 38–41, June 1988.

[53] Mart L. Molle, Khosrow Sohraby, and Anastasios N. Venetsanopoulos. Space-Time Models of Asynchronous CSMA Protocols for Local Area Networks. *IEEE Journal on Selected Areas in Communications*, 5(6):956–968, July 1987.

[54] Geoffrey Gordon. *System Simulation.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1978.

[55] Sun. SunOS Reference Manual. Release 4.0., 1989.

[56] R. Muntz. Analytic Modeling of Interactive Systems. In *Proceedings of IEEE, Vol 63, No. 6*, pages 946–953, June 1975.

[57] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis.* McGraw-Hill Inc., 1982.

[58] David J. Aldous. Ultimate Instability of Exponential Back-Off Protocol for Acknowledgement-Based Transmission Control of Random Access Communication Channels. *IEEE Transactions on Information Theory*, 33(2):219–223, March 1987.

[59] Scott Shenker. Some Conjectures on the Behavior of Acknowledgement-Based Transmission Control of Random Access Communication Channels. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Banff, Alberta, Canada, May 1987.

[60] Johnathan Goodman, Albert G. Greeneberg, Neal Madras, and Peter March. Stability of Binary Exponential Backoff. *Journal of the ACM*, 35(3):579–602, July 1988.

[61] Martin Reiser and Stephen S. Lavenberg. Mean Value Analysis of Closed Multichain Queuing Networks. Technical Report Report RC-7023, IBM T.J. Watson Research Center, March 1978.

[62] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data structures and algorithms.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.