Virtual Time CSMA: Why Two Clocks Are Better than One

MART L. MOLLE, MEMBER, IEEE, AND LEONARD KLEINROCK, FELLOW, IEEE

Abstract—**A new carrier sense multiple access (CSMA) algorithm, called virtual time CSMA, is described and analyzed. This algorithm uses a novel approach to granting access to the shared broadcast channel based on variable-rate clocks. Unlike other CSMA algorithms, the operation of virtual time CSMA reduces to the ideal case in the zero propagation time limit: a work-conserving, first-come first-served $M/G/1$ queueing system. The algorithm does not appear to be difficult to implement, but offers better throughput-delay performance than existing CSMA algorithms. A simple closed form technique for estimating the mean message delay is presented. This technique is of independent interest because of its applicability to certain "sliding window" tree conflict resolution algorithms. Extensive numerical results for the algorithm are presented, including comparisons with simulation and with other CSMA algorithms.**

I. Introduction

ONE simple way to design a communications network is to allow the stations to exchange messages over a shared broadcast channel. The channel is monitored by all the stations, providing a direct communications path between every pair of stations in the network. However, no more than one message at a time can be transmitted successfully over the channel. Whenever the reception of several messages overlap at a station, we say that a *collision* has occurred and assume that none of the transmissions are received correctly by the station. Thus, for the network to operate successfully, the stations must abide by a common set of rules governing access rights to the channel, called a *multiple access protocol*.

The difficulty in designing a good multiple access protocol arises from the spatial distribution of the stations. Since no additional communication paths are provided between the stations,[1] there can be no observable network-wide queue. Thus, no protocol can arrange a perfect schedule for the message transmissions. Instead, the actions of a protocol simply specify a mapping from the time axis into (possibly empty) subsets from the set of all possible waiting messages. A message is transmitted whenever the protocol specifies a subset to which it belongs. Only those subsets may be specified for which each station can determine whether or not it has a message belonging to that subset. Thus, for example, either the oldest remaining message at station N or the set of messages generated between times τ_1 and τ_2 could be specified, but not the oldest remaining message in the entire network. In particular, subset selection can depend on previously selected subsets through observation of the history of activity on the channel.

If the network were to contain few stations (or be very heavily loaded), such fixed scheduling algorithms as round-robin TDMA would work well. Successively inviting each station to transmit any single message clearly could not cause collisions. It would also avoid long idle periods in the presence of waiting messages because there are few stations to examine. Below, however, we consider protocols for networks with very large numbers of stations, each of which rarely generates a message. Here it becomes difficult to find individual stations having a message to transmit, so *random access* protocols that grant access to the channel on a demand basis are more efficient. ALOHA [1] is perhaps the simplest possible random access protocol. Here stations are permitted to transmit new messages at will (or at the start of any *slot* in the synchronous, or "slotted" version [22]). Should a collision occur, each affected station waits for a random time interval and then blindly retransmits its message, and so on.

In this paper, we introduce a new carrier sense multiple access (CSMA) protocol that offers better performance than existing CSMA protocols. CSMA protocols are extensions of ALOHA that take advantage of how short the propagation time across a local network is, compared to a message transmission time. Here stations monitoring the channel can determine whether or not the channel is idle through *carrier sensing* (and thus can *defer* to other stations' transmissions when it is busy, preventing an inevitable collision). Sometimes it is also possible for stations to determine whether ongoing transmissions are colliding (so that the duration of collisions can be reduced) through *collision detection*—see [16].

The operation of CSMA protocols can be either asynchronous (unslotted) or synchronous (slotted). In asynchronous protocols, each station runs its local copy of the protocol independently, using local observations of the evolution of the channel state. In synchronous protocols, all stations run their local copies of the protocol in lock-step. Time is divided into a sequence of *slots*. Transmissions are not allowed to cross slot boundaries: stations can only begin transmitting at the start of a slot; the start of next slot takes place only when it is clear that the reception of all those transmissions (if any) must be complete at every station.

We model collision detection, when it is available, in the following way. Shortly after the start of the first interfering transmission reaches a transmitting station, that station recognizes that a collision is taking place. Thereafter, the station stops transmitting the remainder of its message, briefly "jams" the channel (by transmitting a strong burst of noise) to ensure that all stations consistently recognize the collision, and then remains silent until the channel becomes idle once again. In the asynchronous algorithm, we must follow the model described above exactly: the duration of each colliding transmission is the sum of the transmission time up to the arrival of the first interfering transmission and

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication without oral presentation. Manuscript received January 27, 1981; revised February 21, 1985. This work was supported by an Operating Grant from the Natural Sciences and Engineering Research Council of Canada, and by the Defense Advanced Research Projects Agency under Contract MDA903-77-C-0272.

M. L. Molle is with the Computer Systems Research Institute, University of Toronto, Toronto, Ont., Canada M5S 1A4.

L. Kleinrock is with the Department of Computer Science, University of California, Los Angeles, CA 90024.

[1] Here we assume that stations may observe the outcome of past transmission attempts at no cost, but that no other "free" information exchange is provided. This model corresponds to networks sharing a transponding satellite channel and to local networks in which all stations monitor the channel to detect idle periods and collisions.

the *collision recovery time*, c, a constant that includes the times for collision recognition and jamming the channel. In the synchronous algorithm, we simply choose the duration of slots where collisions take place so that they can handle the worst cast, namely, where the duration of each colliding transmission is some fraction b of the message transmission time.

At present, there are three well-known CSMA protocols [10]. In *nonpersistent CSMA*, only those messages that arrive when the channel is sensed idle are transmitted. This transmission takes place immediately in the asynchronous version, and at the start of the next slot in the synchronous version. All other messages are immediately rescheduled as if a collision had occurred. In the *1-persistent CSMA* protocol, all messages are transmitted as soon as possible. In the asynchronous version, messages are transmitted immediately if they arrive when the channel is idle, or as soon as the channel becomes idle otherwise. In the synchronous version, messages are transmitted at the start of the next slot. The *p-persistent CSMA* protocol is a generalization of 1-persistent CSMA, in which a message that arrives when the channel is busy is transmitted only if the length of the next idle period exceeds a geometrically distributed random value with parameter p.

II. Why Not an Inference Seeking Tree Algorithm?

Before launching into the details of our new CSMA protocol, it may be helpful to explain why CSMA protocols in general are of interest for local networks. Two general classes of random access protocols can be found by examining the reaction of the protocols to the history of events on the channel. We call the first class "inference avoiding" protocols because they try to operate as if the past history of events on the channel did not matter. This class includes the ALOHA protocol [1] and various CSMA protocols [10]—see [26] for a survey. The basic idea here is to simplify the problem by modeling inference avoiding protocols as two separate algorithms: channel access and retransmission feedback. A *channel access algorithm* selects the subsets for transmission as if the distribution of the number of messages in those subsets depends only on the subset specified (e.g., the particular range of generation times), independent of the previous channel history. Typically, the performance of such channel access algorithms is found under the assumption that the total traffic, and hence the distribution of messages in each selected subset, is Poisson. Should some subset selection result in a collision, the channel access algorithm carries on as if the affected messages were lost. If the network is not a loss system,[2] a *retransmission feedback algorithm* (sometimes called a "backoff algorithm") is invoked following each collision to select, independently and at random, a new "generation time" for each of the affected messages. Care must be taken in designing a retransmission feedback algorithm or an inference avoiding protocol may become unstable [13], [4], [3], [28].

We call the second class of random access protocols "inference seeking" protocols (or "tree conflict resolution algorithms," as they are usually called—see [7], [2], [27]). These protocols apply various statistical inference techniques to the channel history information to improve the subset selection process. (The name "tree conflict resolution algorithm" arose because these protocols can be modeled as decision trees.)

It is well known that inference seeking protocols offer some important advantages over inference avoiding protocols in a friendly environment. First, inference seeking protocols can be made inherently stable without the imposition of controls on the feedback of collisions. Second, inference seeking protocols can use the channel more efficiently than can inference avoiding protocols, because they exploit rather than ignore the channel history information in their scheduling decisions. This advantage is most apparent when the channel history information is only made available to the stations after some delay (as is the case in a satellite channel), and declines significantly in local networks where rapid idle- and/or collision-detection is possible—see [17].

In a more hostile environment, such as a mobile packet radio network [8], the behavior of the channel is far from ideal. The *capture* effect allows some stations to correctly receive a strong signal even when it is colliding with a weaker signal. Since signal strength decays with distance, relative signal strength (and, hence, the channel history) is position dependent. A packet radio network may also be partitioned from time to time because the stations are mobile and could pass through a tunnel, say. Finally, atmospheric effects occasionally may render the history information incorrect at some or all of the stations.

Inference seeking protocols can suffer two kinds of failure when the channel history information is unreliable. First, Massey [5] has shown that *consistent but incorrect* information can cause deadlocks when certain inferences are attempted. In his example, a noisy channel made an idle period appear to be a collision—thereby sending the protocol on an endless search for some nonexistent messages. However, this same "missing message" deadlock condition could also occur whenever messages are lost *after* they have been involved in a collision. Such message loss could clearly happen when a station leaves the network (possibly due to failure of the station or through disruption of the channel). However, a more serious cause of this deadlock is the interaction of various protocol layers. After some prearranged "timeout" period has elapsed, it is common for higher level protocols to stop trying to transmit a message.

The performance of inference seeking protocols also suffers when there are *inconsistent* data at the different stations. If the stations are not aware of this situation, coordination among the stations is lost, degrading the performance. If the stations are aware of this situation (say, when a new station is trying to join an operating network, or when a station is trying to rejoin after a channel disruption), those stations that do not know the "correct" state of the protocol may be forbidden from accessing the channel until they can determine it, which could take some time.

It should be clear from the discussion above that CSMA protocols are useful for certain types of local networks (such as packet radio networks) *because* they are inference avoiding and, thus, potentially more robust when faced with misleading, incorrect, or inconsistent channel history information. The study of CSMA protocols is also of importance because of their use in actual networks, such as the Ethernet [16] coaxial cable network and the Department of Defense PRnet packet radio network [8].

III. Description of the Virtual Time CSMA Channel Access Algorithm

In virtual time CSMA, messages are assigned transmission times through a 1-1 mapping that compresses the entire "real" time axis (including the arrival times for all messages) until it fits onto a "virtual" time axis consisting of the union of all idle periods on the channel. To understand how this is done, it is best to start with an example—see Fig. 1. In the initial state of the algorithm (before the first channel busy-time is observed), the transmission time for a message is the same as its arrival time. Eventually the first transmis-

[2] In some applications, such as packetized voice, it makes sense to drop colliding packets, because the systems are more tolerant of the loss of some packets than of either unordered delivery of packets or long and variable packet delays.

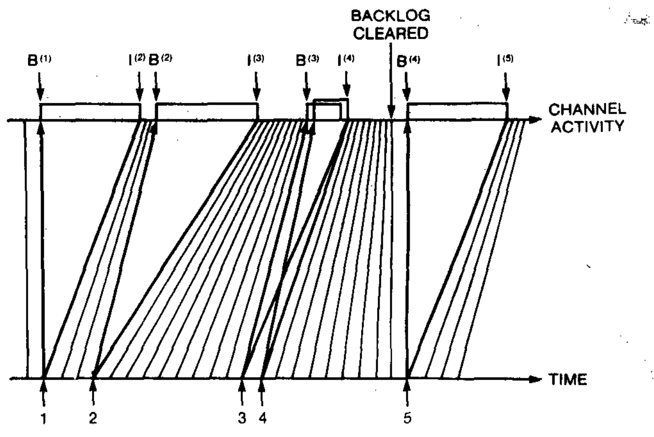


Fig. 1. Compressing the time axis onto the channel idle periods.

sion(s) occur, and thereafter the stations sense the channel busy over some interval, say from $B_j^{(1)}$ to $I_j^{(2)}$ at the jth station.[3] Because of carrier sensing, the algorithm must suspend further transmissions during that busy interval, even though new messages continue to arrive. When transmissions resume at $I^{(2)}$, the *backlog* (i.e., the amount by which transmissions have fallen behind arrivals) has grown to $I^{(2)} - B^{(1)}$. Thus, as the algorithm begins "clearing" this backlog (by transmitting any messages in the network chronologically, starting with those that arrived at $B^{(1)}$), the transmission time for a message has become the sum of its arrival time and the backlog at $I^{(2)}$. Obviously, something must be done to reduce this backlog over time. Otherwise messages arriving after $I^{(k+1)}$ would be delayed by at least $\Sigma_{i=1}^{k} (I^{(i+1)} - B^{(i)})$, which grows without bound as $k \to \infty$. To prevent this, whenever the algorithm is trying to clear a backlog, it *compresses* the time axis by a factor of η, $\eta > 1$. In this case, two distinct points on the time axis, say τ_1 and τ_2, will be mapped into distinct transmission times, τ_1' (during the jth channel idle period) and τ_2' (during the kth channel idle period), respectively, such that

$$\frac{\tau_2 - \tau_1}{\tau_2' - \tau_1' - \sum_{i=j}^{k-1} [I^{(i+1)} - B^{(i)}]} = \eta. \tag{1}$$

Should the algorithm ever clear the backlog completely, it returns to its initial state. Thus, messages are always transmitted when the algorithm has cleared the backlog (if any) up to their arrival times. Note that 1-persistent behavior can be obtained by choosing $\eta = \infty$. For finite η, the backlog is a nonnegative random walk which *increases* at a rate of unity while the channel is busy, and *decreases* at a rate of $\eta - 1$ while the channel is idle.

The virtual time CSMA channel access algorithm is simple to implement, even though, in general, the transmission time for a message can depend on channel activity that takes place both before and after its arrival. Assume that each station is equipped with two "clocks," measuring the passage of "real" time and "virtual" time, respectively. Without loss of generality, both clocks are initialized to zero. Thereafter, the real-time clock runs continuously at constant rate, while the virtual-time clock runs as follows. When the channel is sensed *busy*, the virtual clock stops (retaining its current reading); when the channel is sensed *idle*, the virtual clock is allowed to run (in the manner described below). In the *asynchronous* (unslotted) version, the virtual clock runs continuously at a constant rate, either η times faster than the real-time clock if it has fallen behind the real-time clock, or in lock-step otherwise. In the *synchronous* (slotted) version, all virtual clocks advance by one "step" at the beginning of each slot.[4] Each step advances their readings by the minimum of the current backlog, Q, and the constant $\omega \triangleq a\eta$, so that idle periods, consisting of slots of duration a, are compressed by a factor of η when there is a backlog. Once we set the clocks operating, the rule for transmitting messages becomes obvious. Each message is tagged with the reading of the real-time clock when it arrives, and is transmitted when the virtual clock reading passes the tagged value—see Fig. 2. (Were we to observe the stations in virtual time—i.e., only when the virtual clocks are allowed to run—we would conclude that they were following the ALOHA protocol.) It should be clear that the clocks need not be synchronized between stations. In fact, "real time" can advance at a different rate at every station without affecting the algorithm, as long as the same clock speed ratio is maintained. (The algorithm will still operate if the clock speed ratios vary from one station to another, but stations with higher clock speed ratios will receive higher priority.)

Virtual time CSMA can also be described in terms of a collection of concurrent processes [12]. Assume that each station has available a single "real-time" clock and a programmable interrupt timer; the virtual-time clock is simulated by remembering the most recent value of virtual time and the (real) time at which it was last updated. Stations must also maintain a sorted transmit queue, and record the current state of the channel. The *channel monitor* process runs whenever the state of the channel changes. Whenever the transition from busy to idle channel takes place, the current virtual time reading is marked up to date (since it must have remained stopped since the last update) and a timer interrupt is set to take effect when the first message in the queue would be sent if no other channel activity were observed in the mean time. (It is set to infinity if the transmit queue is empty.) Whenever the transition from idle to busy channel takes place, the virtual clock reading is updated, reflecting the fact that it has been running since the last update. If the transition to busy channel occurred before the timer interrupt (because of some other station's transmission), the timer interrupt is cancelled. The *transmitter* process transmits the message at the head of the transmit queue whenever a timer interrupt occurs. The *scheduler* process runs each time a new message arrives or a colliding message must be rescheduled. Each message is tagged with the current value of real time (plus the retransmission delay in the case of collisions) and inserted into the sorted transmission queue. If the channel is idle and the insertion took place at the head of the transmission queue, the timer interrupt is set to take effect when the newly inserted message should be transmitted. If there is collision detection, a *collision* process is run whenever the station discovers that its current transmission is part of a collision. This process cancels the transmission of the remainder of the message and then jams the channel for a collision recovery time [16].

Virtual time CSMA offers several advantages over other versions of CSMA. Unlike nonpersistent CSMA, every

[3] In general, because signal propagation is not instantaneous, these times (and even their difference in the case of collisions) will vary from station to station. However, their dependence on the identity of the station is unimportant for our example.

[4] In its synchronous version, our "virtual clock" abstraction is equivalent to the "sliding window" abstraction described in the tree algorithms of Gallager [5] and Tsybakov and Mikhailov [29]. However, the virtual clock idea can be generalized to asynchronous operation. (Indeed, in [18] we showed how to construct asynchronous tree algorithms from "virtual clock" descriptions of their operation.)

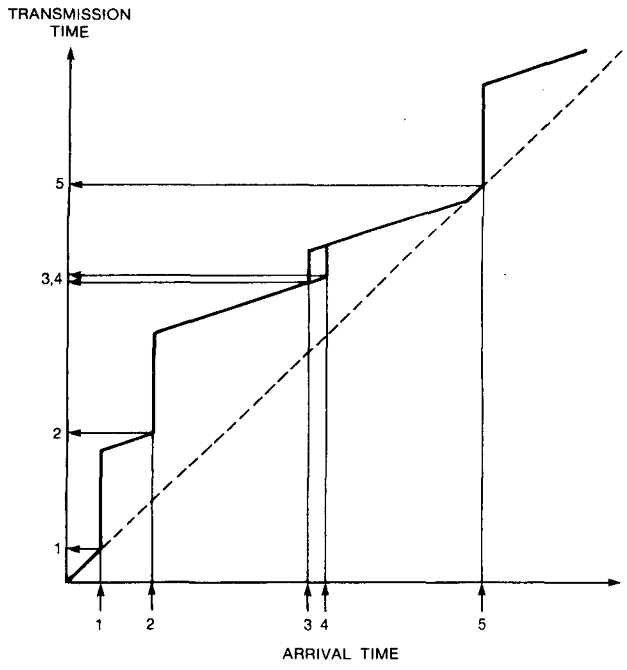


Fig. 2. The mapping from arrival times to transmission times.

arriving message will be transmitted. Unlike p-persistent CSMA, these transmission times are assigned in first-come first-served (FCFS) order. Unlike both 1-persistent and p-persistent CSMA, it is fair in the sense that when a message is transmitted, its probability of success does not depend on its arrival time. In addition, virtual time CSMA is a closer approximation to the "ideal" channel access algorithm, where only an unbroken train of successful message transmissions take place while there are undelivered messages in the network (i.e., work-conserving FCFS single-server $M/G/1$ queueing system). Obviously, this ideal behavior is not realizable because the stations are distributed in space: even though each station is quite capable of managing its *own* queue of messages, finding the status of the queues at *other* stations requires both time (making the information out of date when it arrives) and channel bandwidth (adding overhead to the system). However, in the limit as $a \to 0$, the overhead due to spatial distribution vanishes. Here it should be possible to attain ideal behavior with a channel access algorithm. For the other CSMA protocols described above, this is known not to be the case [24]. For virtual time CSMA, we attain ideal behavior when we let $\eta \to \infty$ as $a \to 0$ such that $a\eta \to 0$. Assume that the aggregate arrival of messages to all stations in the network is Poisson with intensity G per unit time, $0 < G < \infty$. Then the average channel idle time between transmissions is $1/\eta G$ when there is a backlog (which vanishes as $\eta \to \infty$), while the probability that the first transmission after an idle period is successful is at least $a\eta Ge^{-a\eta G}/(1 - e^{-a\eta G})$ in the synchronous version and $e^{-a\eta G}$ in the asynchronous version (both of which converge to unity). Thus, as $a \to 0$, the normalized throughput S converges to min $\{G, 1\}$.

IV. Derivation of the Throughput Equations

To find the throughput equations for virtual time CSMA, we follow the standard assumption that the total traffic (including new message arrivals and retransmissions) can be approximated by a Poisson process with intensity G per unit time. Furthermore, we make the pessimistic assumption that the number of stations is infinite: all messages are certain to be queued at different stations, and hence, stations cannot prevent any collisions by the scheduling of their own transmissions.

We view virtual time CSMA as having two modes of operation. We say that the algorithm is "backlogged" when the virtual clock is allowed to run at rate η, and "caught up" when the virtual clock is allowed to run at rate unity. When the virtual clock is stopped, the mode is determined by the rate from the last time the virtual clock was allowed to run. Thus, to find the throughput equation, we need only find the conditional throughput equations for each mode separately and then average them in proportion to the time spent in each mode.[5]

It is impossible for an observer who is only permitted to monitor the channel when a single mode of operation is in effect (say, corresponding to virtual clock rate r) to distinguish virtual time CSMA with traffic intensity G from nonpersistent CMSA protocol with traffic intensity rG. Both protocols disable new transmissions when the channel is busy (either by blocking messages in nonpersistent CSMA or delaying them in virtual time CSMA). Both protocols enable new transmissions to take place when the channel is idle, and the time until the next transmission begins is exponential with parameter rG. Thus, the conditional throughput equations can be found in the same manner as for nonpersistent CSMA [10].

Under *synchronous* (slotted) operation, the throughput S can be expressed as

$$S = E[H_s]/E[L_s], \tag{2}$$

the ratio of the expected amount of useful work (i.e., the number of successful transmissions) performed in a random slot to the expected duration of a random slot, respectively. Because stations monitor the channel during each slot, the duration of a slot depends on the channel activity in that slot. Idle slots last for the end-to-end propagation time a, so that all stations can be certain that it is idle. In addition to the propagation time, busy slots must also allow for some transmission(s) to take place. Recall that the transmission time for messages is unity. However, because of collision detection, we define the transmission time for colliding messages to be b, $b \leq 1$, the fraction of each message that gets sent before the collision is detected (if at all). Thus, each busy slot lasts for either $1 + a$ if it contains a successful transmission or $b + a$ if it contains a collision.

Because of our Poisson total traffic assumption, the number of messages transmitted in each slot is Poisson. By neglecting the boundary effect described above, the mean of this distribution must be either aG or $a\eta G$, say with equilibrium probabilities π_0 and π_1, respectively. Clearly,

$$E[H_s|rG] = arGe^{-arG} \tag{3}$$

and

$$E[L_s|rG] = ae^{-arG} + (1+a)arGe^{-arG} + (b+a)[1-(1+arG)e^{-arG}] \tag{4}$$

for $r \in \{1, \eta\}$. Thus, to find the throughput in equilibrium, it remains to find $\{\pi_i\}$ so that $E[H_s]$ and $E[L_s]$ can be determined. Assume that the value of η is large enough for the backlog to remain finite with probability one. (Otherwise $\pi_1 = 1$ and message delays would be infinite.) In this case,

[5] Actually, a boundary effect can occur as the virtual clocks catch up to real time. In the synchronous algorithm the clocks could advance by an intermediate value at a step, while in the asynchronous version the clocks could slow down to rate unity during the vulnerable period for some transmission. The probability of success for messages transmitted at such a boundary will be between the values of the two separate modes. It can be shown that ignoring this boundary effect is pessimistic.

$\{\pi_i\}$ can be found by equating the average duration of a slot (i.e., $E[L_s]$) with the average advance of the virtual clock per slot, namely $\pi_0 a + \pi_1 a\eta$. Since $\pi_1 = 1 - \pi_0$, we find

$$\pi_0 = \frac{\min\{0,\ E[L_s|\eta G] - a\eta\}}{E[L_s|\eta G] - a\eta - E[L_s|G] + a}. \quad (5)$$

Fig. 3 shows the throughputs as a function of total traffic for synchronous virtual time CSMA with $\eta = 3$, slotted nonpersistent CSMA, and slotted 1-persistent CSMA. (We assume that $a = b = 0.2$, which corresponds to a 10 km long Ethernet-like local area network where the data rate is 10 Mbits/s, the propagation speed is 200 m/μs, and all packets are 256 bits long [24].) Notice that the curve for virtual time CSMA dominates the curve for 1-persistent CSMA, and that the throughput curve for nonpersistent CSMA also represents the *conditional* throughput for virtual time CSMA, given that $\eta = 1$. Both curves attain the same maximum throughput, but with $\eta = 3$ the peak occurs at one-third the traffic intensity. It should also be clear that as G increases, we can always decrease η to ensure that the product ηG is invariant, and thus attain the same maximum throughput. Although this would result in unbounded message delays in steady state, dynamic adjustments to the virtual clock rate are a "natural" way to control the algorithm under transient overloads.

Under *asynchronous* (unslotted) operation, the throughput can be expressed as

$$S = E[H_c]/E[L_c], \quad (6)$$

the ratio of the expected amount of useful work performed in a random "transmission cycle" (described below) to the expected duration of a random transmission cycle, respectively. Recall that with asynchronous operation, each station runs its copy of the algorithm independently using local observations of the channel state. Thus, the performance of the algorithm is sensitive to such characteristics of the network as the normalized propagation time between each pair of stations, $\{a_{ij}\}$, and the traffic matrix. Here we follow [10] in assuming $a_{ij} = a$, the worst case, for all stations. (This assumption correctly models a "star" topology, e.g., [21].) The details of the traffic matrix are unimportant, as long as we can neglect the probability that a single station transmits several messages in rapid succession. (Recall that we have already made the assumption that the total traffic is Poisson with intensity G.)

We shall describe the operation of the asynchronous algorithm as a sequence of *transmission cycles*. The jth transmission cycle runs from $I^{(j)}$ until $I^{(j+1)}$, consisting of the jth channel idle period followed by the jth channel busy period. Since in continuous time the exact channel history depends on the observation point, we shall always monitor these state transitions from the "hub" of the star.

A key step in the throughput calculation is illustrated in Fig. 4. By neglecting the boundary effect as the backlog is cleared, we can partition the time axis into two virtual time axes, corresponding to virtual clock rates η and unity, respectively. By separately modeling operation of the algorithm over the two virtual time axes, we face a simpler task. Were we to model the operation of the algorithm over real time, we would have had to analyze "multimode" transmission cycles where the virtual clock rate drops from η to unity part way through an idle period. But by following the operation of the algorithm over the virtual time axis at a time, we need consider only "single-mode" transmission cycles where the virtual clocks run only at some fixed rate r, say, during each idle period. This decomposition does not affect the observed behavior of the algorithm over a transmission cycle because of the memoryless property of Poisson arrivals.

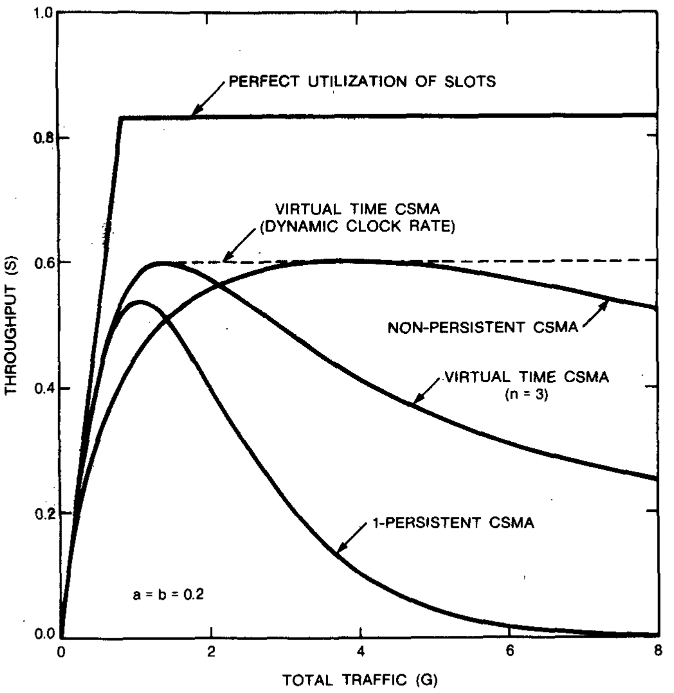


Fig. 3. Throughput curves for synchronous CSMA algorithmns on an Ethernet-like network.

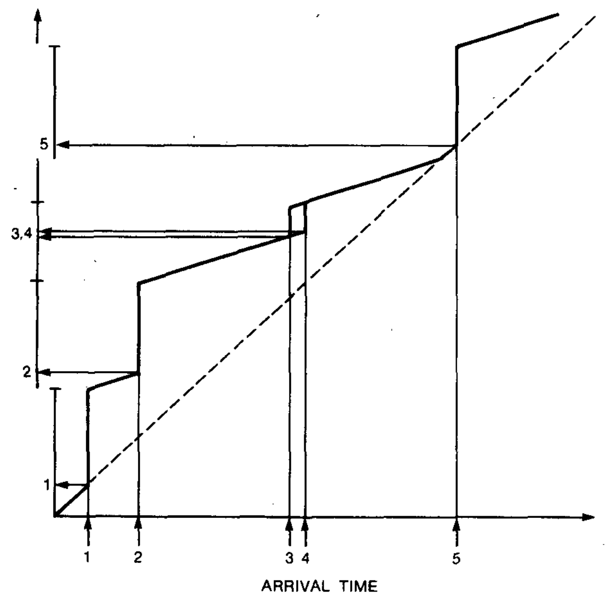


Fig. 4. Partitioning channel activity into two virtual time axes.

In Fig. 4, we have redrawn Fig. 2 to show how the two virtual time axes corresponding to clock rates η and unity, respectively, are interleaved on the real time axis, and how each virtual time axis can be divided into a sequence of "single-mode" transmission cycles. In the $r = 1$ virtual time axis, each transmission cycle consists of a single connected component in real time that begins when the backlog is cleared part way through some idle period and lasts through the following channel busy time (e.g., messages 1 and 5). In the $r = \eta$ virtual time axis, transmission cycles could be split into several components by transmission cycles from the $r = 1$ axis (e.g., 5's transmission cycle separates the beginning of the last $r = \eta$ transmission cycle from the rest of that cycle). Note that a transmission cycle on the $r = \eta$ virtual time axis always includes one complete transmission cycle on the real time axis, and possibly also the beginnings of some earlier channel idle periods up to the point were the backlog was cleared.

Consider a single-mode cycle with virtual clock rate r. At some time t, the cycle begins as the channel changes state from busy to idle. At time $t + a/2$, all stations that had not just been transmitting sense this same state change. Thereafter, the channel remains idle everywhere for an exponential interarrival time (with mean $1/rG$) until the next transmission begins. Following the start of this first transmission, a further time of $a/2$ elapses before we observe the transition from idle to busy at the hub. Note that this first transmission will be successful (and hence, one unit of useful work will be accomplished in this cycle) if no other station subsequently decides to transmit a message before it detects the start of the first transmission. Because of our worst-case star topology, each of the other stations has the opportunity to cause a collision for time a, so that

$$E[H_c|rG] = e^{-arG}. \tag{7}$$

If this cycle contains a successful transmission, the channel will remain busy for one time unit, so that $E[L_c|rG, \text{success}] = 1 + a + 1/rG$. If this cycle contains a collision, the length of the busy period (and, hence, of the cycle) depends on whether or not the stations can detect collisions, but the expected length of the idle period is still $1/rG + a$.

If collision detection is not available, the channel remains busy for the union of the busy periods due to each colliding transmission. Thus, the average busy period duration given that a collision occurred is simply $1 + E[Y]$, where

$$E[Y] = \int_0^a P[Y \geq y]\, dy = \int_0^a \frac{1 - e^{-(a-y)rG}}{1 - e^{-arG}}\, dy$$
$$= \frac{a - (1 - e^{-arG})/rG}{1 - e^{-arG}}$$

is the average time between the beginnings of the first and the *last* colliding transmissions. If collision detection is available, we assume that it operates instantaneously, and thereafter the station stops transmitting the remainder of the message, jams the channel for a *collision recovery time*, c, and then remains quiet until the busy period ends. Once again the busy period at the hub begins $a/2$ time units after the first station to start begins transmitting, and ends $a/2$ time units after the last station to stop ends transmitting. However, for the star topology, it can be shown [12] that the first station to begin transmitting is the last to detect the collision (a time units after the *second* station starts transmitting; all other stations detect the collision a time units after the *first* station starts transmitting) and, thus, also the last station to stop transmitting. Note that the third and subsequent transmissions (if any) have no effect on the duration of a collision busy period. In this case, the average busy period duration given that a collision occurred is $a + E[Y'] + c$, where

$$E[Y'] = \int_0^a \frac{e^{-yrG} - e^{-arG}}{1 - e^{-arG}}\, dy = \frac{(1 - e^{-arG})/rG - ae^{-arG}}{1 - e^{-arG}}$$

represents the average time between the beginnings of the first and the *second* colliding transmissions. It follows that

$$E[L_c|rG] = 1 + 2a + e^{-arG}/rG \tag{8}$$

if collision detection is not available, and

$$E[L_c|rG] = c + 2a + (2 - e^{-arG})/rG + e^{-arG}[1 - 2a - c] \tag{9}$$

if collision detection is available.

To complete the throughput calculation, we must determine $E[H_c]$ and $E[L_c]$. Let π_0 and π_1 be the equilibrium probabilities that $r = 1$ and $r = \eta$, respectively, for a single-mode cycle chosen at random. A virtual clock situated at the hub would run (at rate r) whenever the channel was sensed idle. Since the average idle time in a cycle is $1/rG + a$, the average advance of the virtual clock per cycle is $1/G + \pi_0 a + \pi_1 a\eta$. Equating this with the average duration of a transmission cycle, namely $E[L_c]$, and recalling that $\pi_1 = 1 - \pi_0$, we find

$$\pi_0 = \frac{\min\{0,\ E[L_c|\eta G] - a\eta - 1/G\}}{E[L_c|\eta G] - a\eta - E[L_c|G] + a}. \tag{10}$$

V. Evaluation of Capacity

Having now derived the throughput equations for virtual time CSMA, it remains to show how to find its capacity as a function of the network parameters a, b, and c and the algorithm parameter η. (Because of the Poisson total traffic assumption, we can ignore the effect of the retransmission feedback algorithm on capacity.)

By the *capacity* of a channel access algorithm, we mean the supremum over all attainable values of throughput for which the expected message delay is finite. It is common practice in CSMA to assume that the retransmission feedback algorithm only delays messages for a finite time between transmissions, and, thus, that the capacity of the algorithm is simply the supremum over G of the throughput equation. With virtual time CSMA, however, there is also another constraint limiting its capacity, namely, that the backlog remain finite with probability 1. Thus, to find the capacity of virtual time CSMA, we must restrict G to those values for which $\pi_1 < 1$.

Fig. 5 shows capacity for synchronous virtual-time CSMA as a function of the virtual clock rate η, when $a = 0.01$ and there is no collision detection. The maximum capacity of 0.8655 occurs at $\eta \approx 13.5$, but the capacity remains within 1 percent of this maximum for $10 \leq \eta \leq 20$. Note also that as $\eta \to 100$, the capacity approaches 0.53, which is well known to be the capacity of slotted 1-persistent CSMA [10]. Of course, this is to be expected, since for $\eta \geq 1 + a$, synchronous virtual time CSMA simply enables all the arrivals during the previous slot—which is exactly how slotted 1-persistent CSMA operates.

In Section IV, the throughput of virtual time CSMA with virtual clock rate η for any G was shown to be a convex combination of the throughput for nonpersistent CSMA evaluated at G and at ηG. Thus, it should be clear that for a given set of network parameters, the capacity of virtual time CSMA can never exceed the capacity of nonpersistent CSMA. However, it is also easy to show that there is an optimal value of η, say η^*, for which the capacity of virtual time CSMA is the same as the capacity of nonpersistent CSMA (which is the best possible for inference-avoiding protocols under the Poisson total traffic model—see [17]). To see this, we assume that for the given system, nonpersistent CSMA attains its capacity as $G \to G^0$, namely $S^0 = E[H|G^0]/E[L|G^0]$. But whenever $\eta G = G^0$, the *conditional* throughput for virtual time CSMA, given that the algorithm is backlogged, will also be S^0. Thus, we are done if we can find η^*, satisfying both $\pi_1 < 1$ for all $G < G^0/\eta^*$ and $\pi_1 = 1$ for $G = G^0/\eta^*$, i.e., virtual time CSMA with virtual clock rate η^* attains its capacity (of S^0) as G approaches $G^* = G^0/\eta^*$. Using (5) and (10), it is easy to see that this critical value is

$$\eta^* = E[L_s|G^0]/a \tag{11}$$

for the synchronous algorithm, and

$$\eta^* = (E[L_c|G^0] - 1/G^0)/a \tag{12}$$

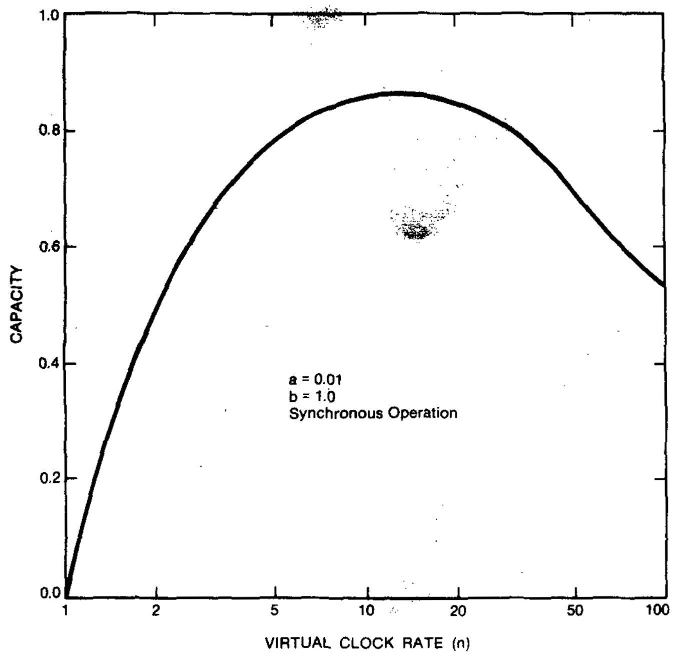


Fig. 5. Capacity as a function of clock rate.

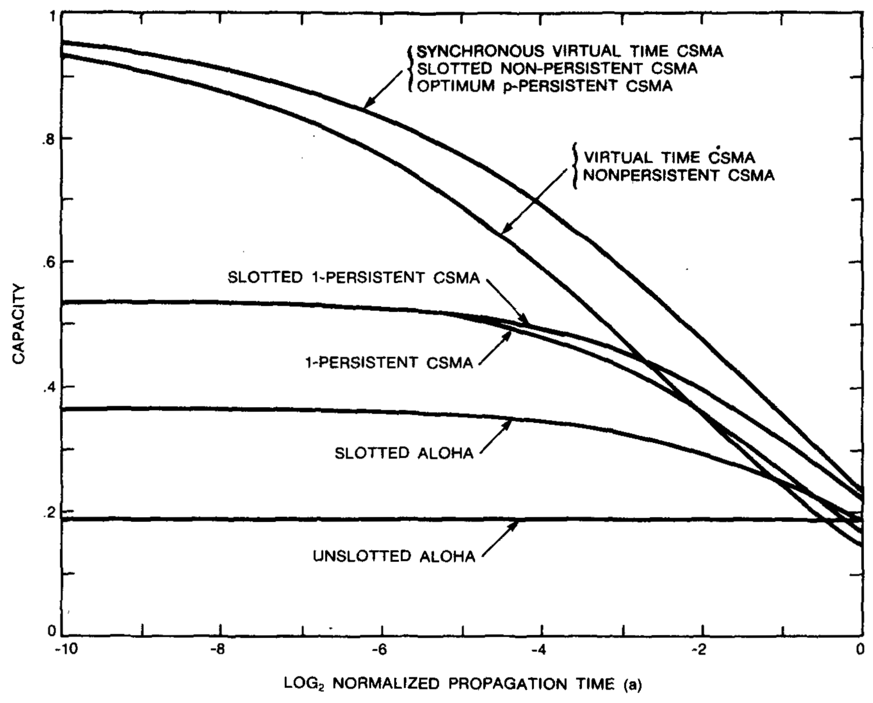


Fig. 6. Sensitivity of capacity to propagation time.

for the asynchronous algorithm. Fig. 6 shows capacity as a function of the propagation time, a, for ALOHA and the various CSMA algorithms.[6]

VI. Estimation of Delay

Here we use the standard equilibrium random access delay model [11] to estimate the mean message delay T. Thus, we assume that

$$T = T_0 + \left(\frac{G}{S} - 1\right) T_R + 1 + a \qquad (13)$$

where T_0 is the *initial delay* from the arrival of a new message until the start of its first transmission attempt (or until the decision is made not to transmit the message in nonpersistent CSMA), T_R is the *retransmission delay* from the start of one unsuccessful attempt to transmit a message until the start of its next transmission attempt, and the term 1 + a represents the transmission time and propagation delay for the message during its final, successful attempt.

[6] Our curves for slotted ALOHA, slotted nonpersistent CSMA, and slotted p-persistent CSMA differ slightly from the corresponding curves in [10]. This is because in slotted ALOHA, they neglected to include the propagation time in the slot length, and in the two CSMA algorithms, the throughput equations were not evaluated precisely at the points where maximum throughput is attained—see [17].

The characteristics of T_0 differ substantially between virtual time CSMA and other CSMA protocols (where T_0 is little more than the residual life of the current slot) because the operation of the virtual clocks causes the messages to queue for access to the channel. Thus, to estimate T, we must first understand how this queueing affects T_0. Fortunately, as will become clear below, performance predictions for virtual time CSMA based on this simple model are surprisingly accurate (and seem much better than the comparable results for the other CSMA protocols). This is because the major cause of the inaccuracy in the model is in the accounting for delay caused by retransmissions, and with virtual time CSMA the expected number of retransmissions per message is small. Thus, unlike other CSMA protocols (where the initial delay is only a small part of the message delay), T_0 is the dominant part of the message delay in virtual time CSMA, and we shall see below that good estimates for T_0 can be found.

Recall that in the limit as $a \rightarrow 0$, the performance of virtual time CSMA approaches the ideal case, namely, a nonpreemptive single-server FCFS $M/G/1$ queue. Thus, our starting point for estimating T_0 will be the well-known $M/G/1$ queue [9]. For any queueing system, we can find the *unfinished work* at time t, $U(t)$, as the sum of the remaining service times of all customers in the system at time t. Assuming that the system is work-conserving, the server performs useful service at the rate of one second of work per second of real time, so that $U(t)$ will tend to decrease towards zero at rate unity. However, each time a new customer arrives, he brings with him a quantity of work equal to his service time, so that $U(t)$ exhibits discontinuities as customers arrive. Hence, the server is *busy* whenever $U(t) > 0$ and *idle* whenever $U(t) = 0$, and if no further customers were to arrive, the server would become idle at $t + U(t)$. Note that U is sometimes called the "virtual waiting time," because a customer arriving at time t would wait for exactly $U(t)$ before entering service, assuming nonpreemptive FCFS scheduling.

In virtual time CSMA, we make use of a generalized definition of unfinished work. Here we say that the unfinished work at time t is the time required to completely clear the backlog present in the system at t, assuming no further customers arrive after t. Note that here we define "customer" to mean a set of one or more messages that are transmitted concurrently (creating a continuous period of channel activity); the arrival time of a customer is the arrival time of the *leading message* in the set. (The arrival times for any other messages are simply ignored.) We further define the "work" that must be expended on such a customer, X, to consist of two components, namely a *resting* component X_R, during which the virtual clocks remain stopped because the channel is busy, and a *scanning* component X_S, during which the virtual clocks are running at a rate of η in an effort to regain the time lost while they were resting. In the asynchronous case, the virtual clocks are stopped for exactly the time that the channel is sensed busy, so that X_R is simply equal to the transmission time for the message(s). To find X_R in the synchronous case, we refer to Fig. 7. At the beginning of a busy slot, the virtual clocks make an *initial advance* of ω that triggers the start of the transmission(s), which we shall account for by defining the first a time units of each slot as scan time. Since the virtual clocks never advance in the middle of a slot, the remainder of the slot must be resting time. Hence, $X_R = L_s - a$. But the duration of each slot always exceeds the transmission time for any message(s) transmitted in that slot by a, to account for the propagation time. Thus, we have once again that X_R is equal to the transmission time. In either case, the virtual clocks must regain this lost time of X_R during the scan time. It is easy to see that whenever the virtual clocks are running at rate η, the backlog must be decreasing at the rate of $\eta - 1$, so that

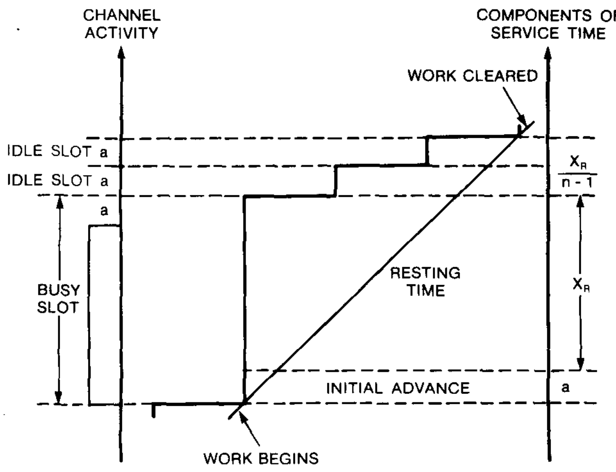


Fig. 7. The work associated with a transmission in synchronous virtual-time CSMA.

$$X_s = \frac{X_R}{\eta - 1} \tag{14}$$

and, hence, that

$$X = X_R \frac{\eta}{\eta - 1}. \tag{15}$$

In Fig. 8, we illustrate the evolution of unfinished work in virtual time CSMA. A new busy period begins at $t^{(1)}$ with the arrival of the first "customer," and the unfinished work increases by $X_R^{(1)} + X_S^{(1)}$. Because the system is empty initially, this customer enters service immediately. The resting (i.e., transmission) component of his service will be completed at $t^{(1)} + X_R^{(1)}$; one propagation time later, the corresponding messages either leave the network if they were transmitted successfully, or else begin their retransmission delay. If no further arrivals were to take place, the scanning component of his service would be completed at $t^{(1)} + X_R^{(1)} + X_S^{(1)}$, ending the busy period. Suppose, however, that a second customer arrives at $t^{(2)}$. Because the virtual clocks start running again at $t^{(1)} + X_R^{(1)}$ (and in virtual time CSMA, this message will be transmitted when virtual time reaches $t^{(2)}$), the resting component of the second customer's service will preempt part of the scanning component of the first customer's service. In Fig. 8, for example, the second customer begins his resting component of service approximately halfway through the scanning component of the first customer's service, while the third customer begins his resting component of service near the end of the scanning component of the second customer's service.

Define the *synthetic queueing problem* to be an ordinary nonpreemptive FCFS queueing system that is given the identical sequence of customer arrival times and service times (including both resting and scanning components) as our virtual time CSMA system. It should be clear from Figs. 8 and 9 that the unfinished work in both systems is identical over all time, and hence, that the virtual time CSMA system and the synthetic queueing problem will have an identical sequence of busy periods. Below we make some key observations about the relation between the waiting times in the synthetic queueing problem and T_0 in our virtual time CSMA system.

Consider the times $\tau^{(j)}$ and $\tilde{\tau}^{(j)}$, measured with respect to the start of a busy period, at which the jth customer enters service in the synthetic queueing problem and the virtual time CSMA system, respectively. In the synthetic queueing

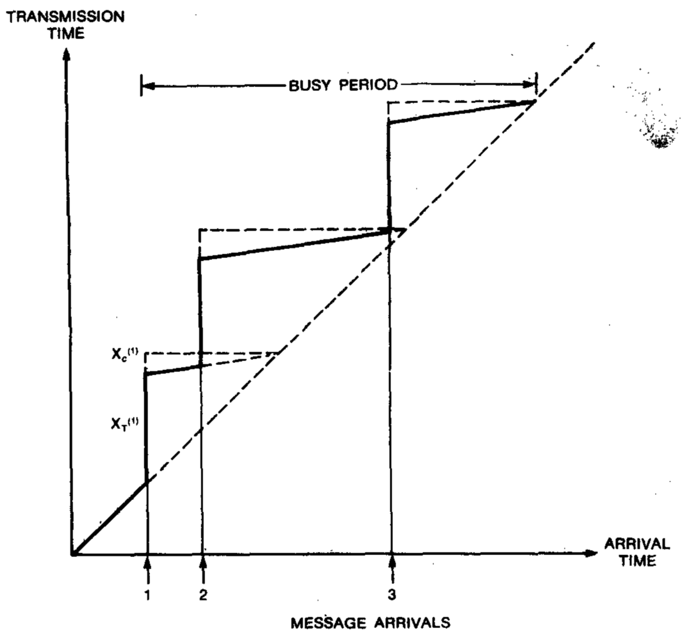


Fig. 8. A busy period in virtual-time CSMA (note how $X_T^{(2)}$ preempts part of $X_C^{(1)}$).

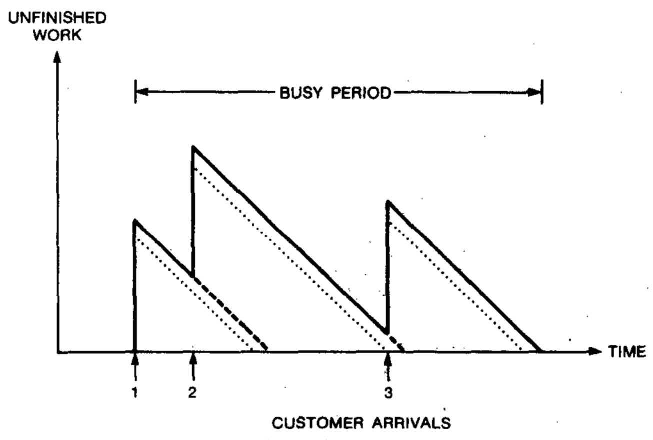


Fig. 9. The corresponding busy period in the synthetic queueing system.

problem customers are served nonpreemptively in FCFS order, so that

$$\tau^{(j)}=\sum_{i=1}^{j-1} X^{(i)}=\sum_{i=1}^{j-1} [X_R^{(i)}+X_S^{(i)}]. \tag{16}$$

In the virtual time CSMA system, the jth customer must still wait until the resting component of service for customers 1, $\cdots, j-1$ is completed, but he will preempt some scan time because his transmission begins as soon as the virtual clocks advance to his arrival time. Thus, assuming that he arrived $t^{(j)}$ after the start of the busy period, he will have preempted enough scan time to advance the virtual clocks from $t^{(j)}$ to $\tau^{(j)}$, namely

$$Z^{(j)} \triangleq [\tau^{(j)}-t^{(j)}]/\eta.$$

It remains to derive a more usable expression for $Z^{(j)}$.

Suppose we find that $\tau^{(i)} < t^{(j)} \leq \tau^{(i+1)}$, for some $1 \leq i < j$. Then we know from the way that the synthetic queueing problem was constructed that, in addition to the sum of the resting components of service for customers 1, 2, $\cdots$, i, $\tau^{(i)}$ also includes enough scan time to advance the virtual clocks to $\tau^{(i)}$, a little less than that required for the virtual clocks to reach $t^{(j)}$. Thus, being careful not to omit the resting components of service for any of the customers 1, $\cdots, j-1$, we see that

$$\tilde{\tau}^{(j)}>\tau^{(i)}+\sum_{k=i}^{j-1} X_R^{(k)}.$$

Similarly, we have

$$\tilde{\tau}^{(j)}\leq\tau^{(i+1)}+\sum_{k=i+1}^{j-1} X_R^{(k)}.$$

It should now be clear that

$$\sum_{k=i}^{j-1} X_S^{(k)}>Z^{(j)}\geq \sum_{k=i+1}^{j-1} X_S^{(k)} \tag{17}$$

and, indeed, we could get the *exact* result for $Z^{(j)}$ by accounting for the time to scan across the residual service time for customer i at customer j's arrival. The final step is to observe that since these results relating the waiting times in each system hold for each customer taken individually, the same results must also hold for the average waiting time in the two systems. Thus, the average waiting time in the virtual time CSMA system, $\tilde{W}$, is obtained from the average waiting time in the synthetic queueing problem, W, by subtracting off $E[Z]$, where Z is the sum of the scan times for every customer in queue on its arrival plus the time to scan across the residual service time of the customer in service.

Now suppose that the synthetic queueing problem were of type $M/G/1$. Then the mean waiting time W is given by the well-known Pollaczek-Khinchin mean value formula. Using Little's result, the number in queue, N_q, can also be expressed in terms of W as $N_q = W\rho/\bar{X}$. Similarly, it is well known [9] for the $M/G/1$ queue that $W = W_0/(1-\rho)$, where W_0 is the mean residual service time for the customer (if any) found in service by a new arrival. Recalling that any customers found in queue by a new arrival to an $M/G/1$ queue will have ordinary service time requirements, we see that if the synthetic queueing problem were of type $M/G/1$, then the waiting time in the virtual time CSMA system would be given by

$$\tilde{W}= W-\frac{1}{\eta}(W_0+N_q\bar{X})= W-\frac{1}{\eta}(W(1-\rho)+W\rho)$$

$$= W\frac{\eta-1}{\eta}. \tag{18}$$

It is interesting to note the symmetry in this last result: $\tilde{W}$ is obtained by *inflating* the transmission time for each message by a factor of $\eta/(\eta-1)$ to account for the scan time overhead, and then *deflating* the resulting waiting time estimate by the reciprocal of that factor to account for the preemption of some of that scan time by subsequent transmissions. To use this result, it remains to show how to map the synthetic queueing problem onto an $M/G/1$ queue.

For *synchronous* (slotted) virtual time CSMA, we find T_0 from the *discrete time* $M/G/1$ queue, with ω as its elementary time unit. Here we restrict all state changes to a sequence of equally spaced *arrival points* 0, ω, 2ω, 3ω, $\cdots$. At each arrival point, either no customer or exactly one customer arrives, independently and with probabilities $(1-p)$ and p, respectively. Service times are independent with mean $\bar{X}\omega$ and squared coefficient of variation C_b^2, and each one is a multiple of ω (so that departures can only occur at 0, ω, 2ω, 3ω, $\cdots$). It can be shown [17] that the mean waiting time in a discrete time $M/G/1$ queueing system with utilization $\rho = p\bar{X}$ is given by

$$W = \bar{X}\omega \frac{\rho(C_b^2+1)-p}{2(1-\rho)}. \tag{19}$$

The estimate for $\tilde{W}$ obtained from (18) and (19) can be made exact, given the Poisson total traffic assumption, in the following case. First, we assume that all service times (as inflated to account for the scan time overhead) are evenly divisible by the window size ω. Second, we impose the following modification to the protocol. Whenever the backlog Q is less than ω at the beginning of a slot, we shall insert a "rest period" equal to $\omega - Q$ so that the virtual clocks advance by *exactly* ω at *every* step (or, equivalently, that all windows are forced to be the same size in a sliding window tree algorithm). Obviously, this constant advance restriction can only increase the delay compared to the original system, because messages arriving when the backlog is small must now wait for it to grow to ω before being transmitted. However, this modification drastically simplifies the analysis by ensuring that both the arrival process and the service times are independent of the state of the algorithm. In this case, an arrival point occurs once every ω time units, a nonempty set of messages may arrive at each such arrival point with probability $p = 1 - e^{-\omega G}$, and if so, the service time (in units of ω) for that set of messages is independently chosen to be either $1/(\omega - a)$ with probability $\omega G e^{-\omega G}/(1 - e^{-\omega G})$ (representing a success) or $b/(\omega - a)$ with probability $1 - \omega G e^{-\omega G}/(1 - e^{-\omega G})$ (representing a collision). Thus, recalling that a message must wait $\omega/2$ on average from its actual arrival time to the next arrival point, we find that T_0 for this modified version of synchronous virtual-time CSMA is given by

$$T_0 = \frac{\omega}{2} + \tilde{W} = \frac{\omega}{2} + \frac{W(\eta-1)}{\eta} \tag{20}$$

where W is given by (19).

Without collision detection, it is no more difficult to find T_0 for the ordinary version of the algorithm, in which the virtual clocks advance by min $\{\omega, Q\}$ at the beginning of each slot. To see this, we observe that without the constant advance restriction, collisions would be less likely to occur in the first slot of each busy period than in the rest. But without collision detection, the service time for the first customer in each busy period is no different from the rest, since the length of *every* busy slot is always $1 + a$. Thus, the requirement of i.i.d. service times in the $M/G/1$ model is satisfied. Furthermore, it is clear that once a busy period has begun, the virtual clocks always advance by exactly ω at each slot for the remainder of the busy period. In other words, for a the busy period beginning at t_1, we see that the arrival points that are included *within that busy period* occur at t_1, $t_1 + \omega$, $t_1 + 2\omega$, $\cdots$. Thus, $\tilde{W}$ can be found using exactly the same discrete $M/G/1$ model as before. However, the virtual clocks now scan each idle period in steps of a once every a time units, instead of in steps of ω once every ω time units, reducing the average waiting time until the next arrival point is reached from $\omega/2$ to $a/2$ for messages that arrived during an idle period. Since such messages make up a fraction $1 - \rho$ of the total, T_0 for ordinary synchronous virtual time CSMA without collision detection is given by

$$T_0 = \frac{\rho\omega + (1-\rho)a}{2} + \frac{W(\eta-1)}{\eta} \tag{21}$$

where W is given by (18) and (19), assuming $b = 1$.

Note that these simple results were only achieved because we deliberately defined service times in the synthetic queueing problem to include scan times. Had we instead let each slot (idle or not) be a "customer" who requires one slot's worth of service, then we would have been forced to use a instead of ω as the elementary time unit because the service time for an idle slot is clearly a. It follows that *even with the constant advance modification*, we would still have been faced with analyzing a system where *periodic* customer arrivals occur once every η elementary time units, rather than a system with *memoryless* arrivals—which is a far more difficult task.

For *asynchronous* (unslotted) virtual time CSMA, we can estimate T_0 using the results for the *continuous time* $M/G/1$ queue. Recall that in our throughput analysis, we have already assumed that the total traffic is Poisson, that the number of stations is large, and that the network topology is the worst possible (i.e., the propagation time between all pairs of stations is the same). Unfortunately, even for this case we cannot use (18) without making some further approximations, because the customer arrival process in the synthetic queueing problem is *not* Poisson. First, "gaps" occur in the customer arrival process. Recall that during a transmission cycle where the virtual clock rate is r, the virtual clocks at almost all the stations advance by the same amount,[7] namely, the sum of an exponential interarrival time with mean $1/G$, which advances the virtual clocks to the arrival time of leading message, and the "vulnerable period" for that message, ar. Any messages that arrive during the vulnerable period do not become customers in the synthetic queueing problem. They simply collide with the leading message, possibly changing his service time. Second, whenever the backlog is cleared, the virtual clock rate (and, hence, the duration of the vulnerable period) is reduced. Thus, we see a dependence between the state of the algorithm and the arrival process, since both the mean time between customer arrivals (i.e., $1/G + ar$) and their service times depend on r.

To make use of (18), we shall approximate the arrival process to the synthetic queueing problem by a Poisson process even though the "gaps" occur. Because the gaps are small, few messages fail to become customers in the synthetic queueing problem because of this approximation. Furthermore, our model of the arrival process in the virtual time CSMA system is *already* an approximation (because we said that the total traffic, including retransmissions, is Poisson), so the additional error from assuming Poisson arrivals in the synthetic queueing problem does not seem significant. To handle the state dependence in the mean interarrival times, we form upper and lower bounds on W by assuming that the mean interarrival time is state independent and given by $1/G + a$ and $1/G + \eta a$, respectively. Recalling that the shorter value for the mean interarrival time should be used between the first and second customer arrivals in a busy period, and that the longer value should be used between each of the subsequent arrivals in the busy period, we expect the actual waiting to approach the upper bound under light traffic (since few busy periods contain more than two customers) and the lower bound under heavy traffic (since customers initiate new busy periods). Together, these two approximations allow us to bound $\tilde{W}$ at any value of G, by solving the Pollaczek-Khinchin mean value formula [9] at utilizations of $\rho_U = \bar{X}G/(1 + aG)$ and $\rho_L = \bar{X}G/(1 + a\eta G)$, respectively, i.e.,

$$W_L = \frac{\rho_L \bar{X}(C_b^2+1)}{2(1-\rho_L)} < W < \frac{\rho_U \bar{X}(C_b^2+1)}{2(1-\rho_U)} = W_U. \tag{22}$$

[7] The only exception is at the transmitting stations during a collision. However, this is unimportant in the calculation of T_0 because we have assumed that the number of stations is large and, thus, that these same stations are unlikely to transmit more messages in the same busy period.

Since an arriving message need not wait for the next arrival point in asynchronous virtual time CSMA, T_0 satisfies

$$W_L \frac{(\eta-1)}{\eta} < T_0 < W_U \frac{(\eta-1)}{\eta}. \tag{23}$$

We have now found expressions for T_0, namely, (20) and (21) for the synchronous case and (23) for the asynchronous case. The throughput S was shown in Section IV to be $E[H]/E[L]$. Thus, to find the mean delay T from (13), it remains to find an expression for T_R.

Recall that T_R is composed of an unsuccessful transmission of duration

$$T_F = \begin{cases} b+a & \text{synchronous case} \\ 1+E[Y] & \text{asynchronous without collision detect} \\ a+E[Y']+c & \text{asynchronous with collision detect} \end{cases} \tag{24}$$

followed by the time it takes for the virtual clocks to advance by the scheduled retransmission delay. We assume that the retransmission feedback algorithm operates by adding a random delay to the current value of the arrival time "tag" for each colliding message (which clearly must have been equal to the current value of virtual time), and that these random delays are independently drawn from a common distribution with mean R.

To calculate the mean time for the virtual clock to advance by R, we make use of the fact that $R \gg 1/(\eta - 1)$ must hold for the Poisson total traffic assumption to be reasonable. In this case, we can assume that this advance is distributed over each virtual time axis in proportion to their relative lengths, and it follows immediately from the discussions leading up to (5) and (10) that

$$T_R = T_F + R \tag{25}$$

if the queueing delay due to the virtual clocks is finite (i.e., $\pi_0 > 0$); otherwise we have

$$T_R = \begin{cases} T_F + \dfrac{RE[L_s|\eta G]}{\eta a} & \text{synchronous case} \\ T_F + \dfrac{RE[L_c|\eta G]}{1/G+\eta a} & \text{asynchronous case} \end{cases} \tag{26}$$

if $\pi_0 = 0$.

VII. Numerical Results

Mean message delays as a function of throughput were found numerically for both the synchronous and asynchronous versions of the protocol. In each case, we have compared the results of our analysis with simulations where we have relaxed many of the assumptions used in the analysis. The results show that in spite of its simplicity, our analytical model is remarkably accurate.

For the *synchronous* protocol, we set the parameters in our model to be compatible with the well-known results of Tobagi and Kleinrock [10]. Thus, we assume that messages are of constant length, that the transmission time for a message is unity, that the end-to-end propagation time a is 0.01, and that there is no collision detection (i.e., $b = 1$).

In Fig. 10, we compare simulation results with the estimated delay from (13), (21), and (26). Following [25, ch. 6], the simulation consists of 50 identical stations, each with geometric message interarrival times with mean a/σ and geometric retransmission delays with mean a/ν. Three values for ν, namely 0.0003, 0.001, and 0.003, and a fixed virtual clock rate of $\eta = 12$ were used in this experiment. Each point is based on a run length of 500 000 slots. Because

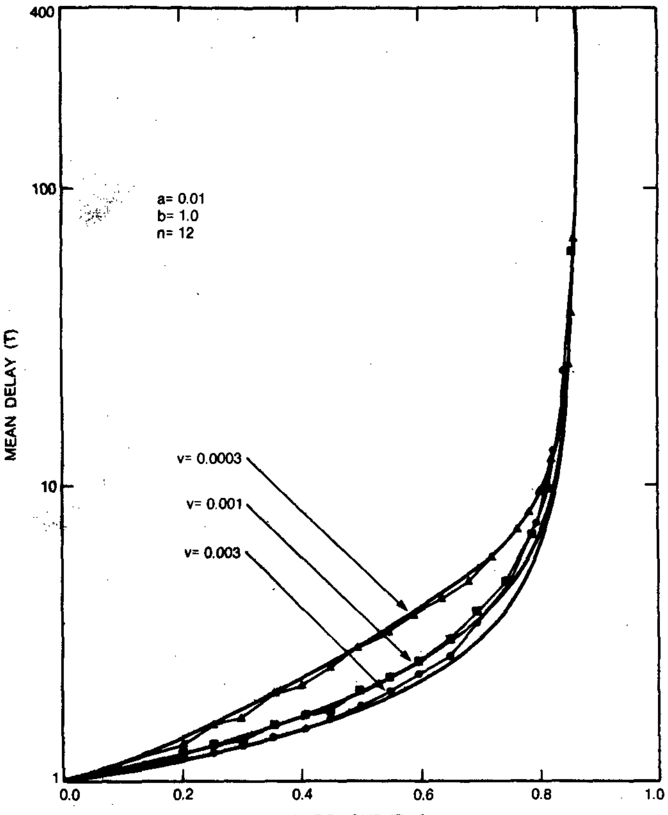


Fig. 10. Comparison of simulation with analytic results for synchronous virtual-time CSMA.

we needed the assumption in our analysis that the total resting and scanning times for each customer, $\eta/(\eta - 1)$ from (15), is evenly divisible by the advance of the virtual clocks in one step, namely ηa, we used $\eta = 12.\dot{1}$ in the analytical curves so that $1/(\omega - a) = 9$.

The agreement between each set of simulation points and the corresponding analytical curve is quite remarkable, considering the simplicity of the analytical model. Indeed, the only noticeable discrepancy is that the simulation points for the higher retransmission rates (i.e., $\nu = 0.001, 0.003$) lie above the corresponding analytical curve, near the "knee" of the delay curve. This appears to be the result of comparing analytical curves in which we made the Poisson total traffic assumption, with simulation results where we used a nonadaptive retransmission feedback algorithm. When the mean retransmission delay is large, we see excellent agreement over the entire throughput versus delay curve. However, when the mean retransmission delay is small, the feedback of colliding messages in the simulation model becomes significant, so that our analytical model underestimates G during the busy periods and, hence, overestimates the probability of success at each transmission attempt.

In Fig. 11, we compare the simulation points for $\nu = 0.003$ and $\nu = 0.001$ with some equilibrium throughput-delay curves for a model of slotted nonpersistent CSMA that was first studied by Tobagi [25]. However, where Tobagi relied on the numerical solution of matrix equations, in the Appendix we solve for the equilibrium solution to the embedded Markov chain as a triangular set of equations, and obtain closed-form expressions for the remaining quantities of interest. This has allowed us to produce more complete performance curves for the comparison. Comparing the simulation points for virtual time CSMA with the equilibrium curves for nonpersistent CSMA in Fig. 11, it is evident

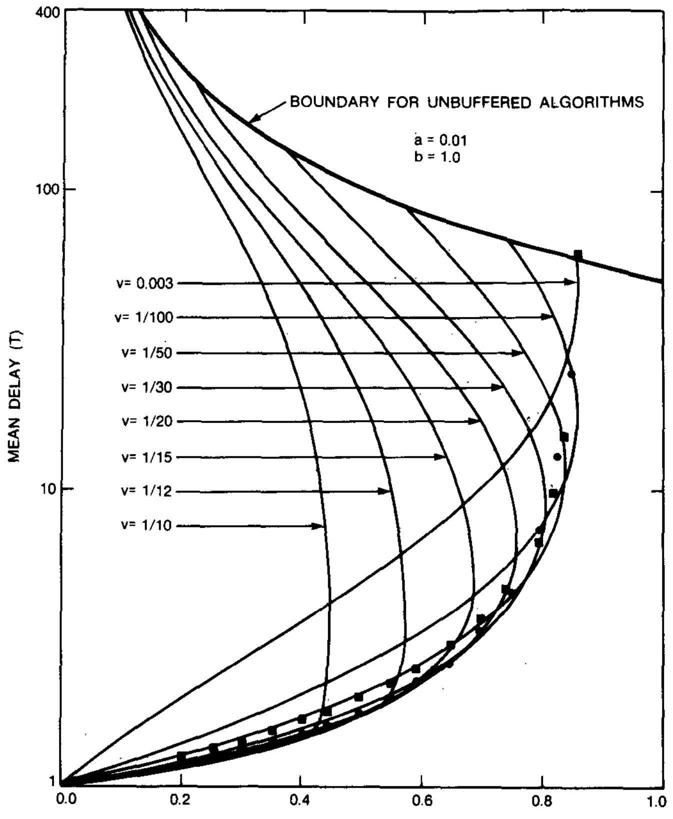


Fig. 11. Comparison of simulation with numerical curves for slotted nonpersistent CSMA. (Simulation of virtual-time CSMA uses $\nu = 0.001$ and $\nu = 0.003$; $\eta = 12$.)

that virtual time CSMA with *fixed* values of virtual clock speed and retransmission delay (i.e., $\eta = 12$, $\nu = 0.003$) attains the lower envelope of the curves for nonpersistent CSMA *optimized* over all values of the retransmission delay.

It is worth pointing out that slotted nonpersistent CSMA has an unfair advantage in this comparison because it assumes *unbuffered* stations: if a station generates a new message before an earlier message is transmitted successfully, the newly generated message is simply dropped without penalty. However, the simulation of synchronous virtual time CSMA assumes stations have *infinite buffers*: all messages are queued at the station until the virtual clocks reach their generation times.[8] Since queueing messages, rather than dropping them without penalty, can only *increase* the mean delay for the messages that are *not* dropped, we expect virtual time CSMA to outperform buffered nonpersistent CSMA.

Before leaving Fig. 11, it is worth remarking that with 50 unbuffered stations, the average number in system, $\bar{N}$, can never exceed 50. It follows from Little's result [14] that it is impossible for the delay curves for nonpersistent CSMA to enter the region bounded below by the curve

$$T_{\mathrm{CRIT}}(S) = 50/S. \tag{27}$$

It is interesting to note that with nonpersistent CSMA, the delay curves for all values of ν actually seem to terminate at some point on this boundary, rather than approaching it asymptotically as $T \to \infty$. Furthermore, each of these limit points represents the capacity of the protocol with *infinite* buffering, given that each station with a nonempty buffer transmits the message at the head of its buffer with probability ν in each slot. To find the capacity of such a buffered protocol, we can use the results of Tsybakov for slotted ALOHA [28] to show that the M-dimensional (embedded) Markov chain representing the queue lengths at each station is ergodic only if the queue at a single station is ergodic, assuming the worst-case behavior at every other station, namely, that their buffers never empty. To see that the throughput of the unbuffered protocol approaches the capacity of the corresponding buffered protocol as its delay approaches the limit point, we note that even for relatively small values of σ, say $\sigma \gtrsim 0.01$, it is very unlikely that a station will remain in the thinking state through an entire transmission cycle (where it will have had more than 100 opportunities to generate a new message). Thus, there will usually be 50 messages in the system, and the channel activity is close to that of the corresponding buffered protocol. The only difference between the buffered and unbuffered protocols occurs after successful transmissions, where the channel load is different because one station is transmitting with probability σ instead of ν. However, this difference in load is not significant as long as $49\nu \gg \sigma$, which was the case in all the above examples.

Figs. 12 and 13 show how our analytical delay model for *asynchronous* virtual time CSMA compares with simulation data.[9] Both simulations consist of 20 stations on a "star" network, each with buffer size 15. The arrival of new messages at all stations is Poisson with identical rates. In Fig. 12 we assume that $a = 0.01$, that the virtual clock rate depends on G through the relation $\eta = 9.45/G$ (since maximum conditional throughput occurs at $\eta G \approx 9.45$), that retransmission delay is geometric with a mean of 3, and there is no collision detection [12, Fig. 6.1]. In Fig. 13 we assume that $a = 0.01$, that the virtual clock rate is fixed at 10, that there is collision detection with collision recovery time 0.001, and that one of three widely varying retransmission feedback algorithms is used [19, Fig. 5]. Since only the random geometric feedback algorithm was simple enough to have an obvious mean, it was used to determine R for the analytical curves.

In both figures we find that the simulation results approach the upper bound to delay at light traffic and the lower bound to delay under heavy traffic, as we expected. Furthermore, the simulation points generally fall between the analytical bounds with two exceptions. First, Fig. 12 shows the same behavior as Fig. 10, where the simulated delays exceed the upper bound near the knee of the curve because of the Poisson traffic assumption. (The fact that the simulation results for binary exponential backoff and the asynchronous stack fall below the lower bound in Fig. 13 is not significant, because the given value of R only applies to the random geometric feedback algorithm.) Second, the simulations attain slightly higher throughput near saturation, where buffer overflow begins to occur. The explanation for this discrepancy lies in the fact that the simulation has a finite number of stations, each with a limited buffer capacity. The inevitable buffer overflow that occurs at saturation interacts with the virtual clock mechanism in an interesting way. Suppose that all stations' buffers are full between times τ_1

[8] To simplify the simulation, a few messages are still lost because no station is allowed to transmit more than one message in the same slot. This could happen if one message suffered so many collisions that it was still in the system at the next message's scheduled transmission time. To prevent this, the new message is dropped without penalty. Such dropped packets are rare, however, because the average number of slots between the first transmissions of two successive messages at the same station is $50/(\omega S)$. In this example, we have $\omega \approx 0.12$ and $S \leqslant 0.86$, so that even at capacity these transmissions are separated by about 475 slots on average.

[9] Many additional simulation experiments involving asynchronous virtual time CSMA, including comparisons between different retransmission feedback algorithms and detailed comparisons with Ethernet-like algorithms on a "bus" topology, can be found in [12], [19].

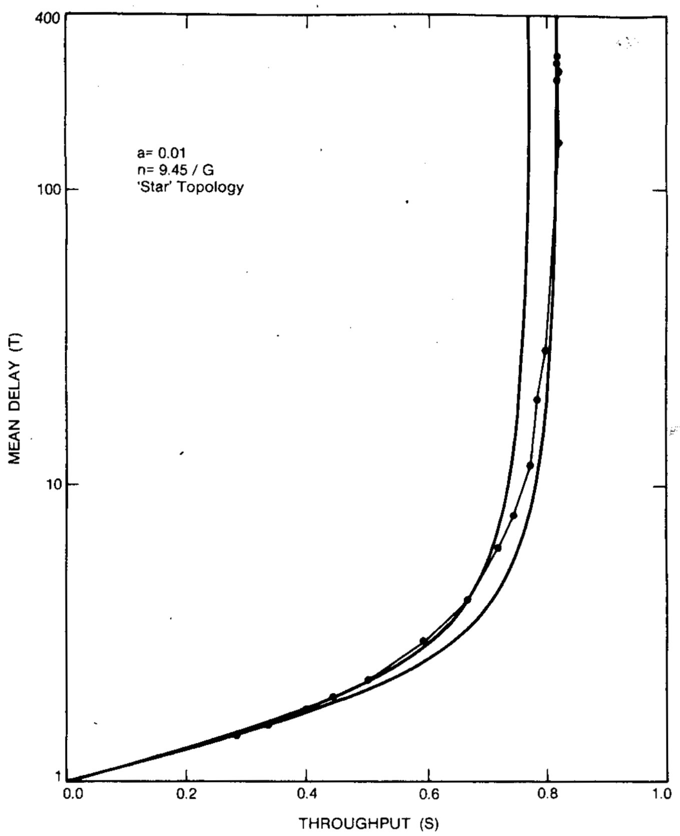


Fig. 12. Comparison of simulation with analytic bounds for asynchronous virtual-time CSMA without collision detection.

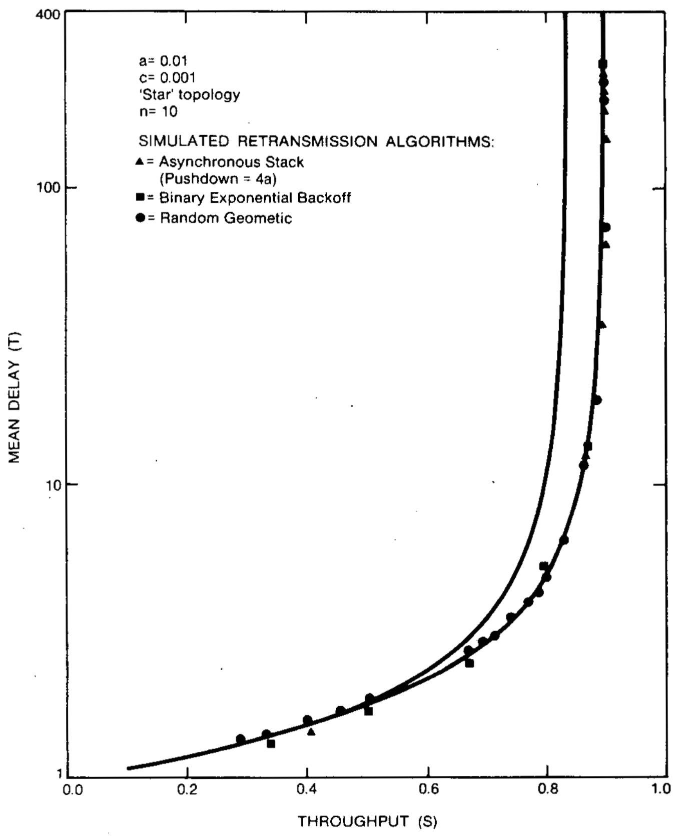


Fig. 13. Comparison of simulation with analytic bounds for asynchronous virtual-time CSMA with collision detection.

and τ_2. Then all new messages arriving in this interval are lost. When the next successful transmission occurs, say by station j, only station j can accept a newly generated message. Thus, later on as the virtual clocks are sweeping past τ_2, only station j will come upon a new message to send, and it is more likely to be able to transmit it successfully than one would expect from the Poisson total traffic model.

VIII. Conclusions

We have introduced the virtual time CSMA channel access algorithm. It is quite different from existing CSMA algorithms, and offers some significant advantages. It is the only CSMA algorithm that reduces to ideal $M/G/1$ behavior in the limit as $a \rightarrow 0$. It is fair in the sense that at each transmission attempt, stations are granted the opportunity to transmit their messages in FCFS order. Its capacity is at least as the other CSMA algorithms, and its delay characteristics are very good. Its virtual clock mechanism is equivalent to the "sliding window" mechanism used in some tree conflict resolution algorithms, but also generalizes to asynchronous (unslotted) operation.

This analysis has also brought to light several items of independent interest. A classification scheme for random access protocols was given, indicating some of the reasons why CSMA protocols are still important in spite of the invention of tree conflict resolution algorithms. The throughput calculation uses a novel decomposition of the time axis into an interleaved set of "virtual" time axes. The method of calculating the mean delay by transforming the system into a synthetic queueing problem and then transforming the results back to the original system has recently been used to find the *exact* throughput-delay curve for certain "sliding window" tree conflict resolution algorithms [20]. We extended Tobagi's analysis of unbuffered slotted nonpersistent CSMA to investigate its behavior at saturation. Finally, we showed how buffer overflow can improve the heavy traffic performance of virtual time CSMA.

CSMA channel access algorithms can be viewed as scheduling algorithms in the sense that they coordinate the use of some common resource (the channel) by a set of competing users (the stations). (Channel access is actually a more difficult problem than scheduling, since the algorithms must operate in a distributed manner, with each station running its own copy of the algorithm without complete information about the demands for service by the other stations.) Sevcik [23] has observed that the difficulty in analyzing a scheduling algorithm seems inversely proportional to the goodness of that scheduling algorithm. Our experiences with virtual time CSMA add to the supporting evidence to this conjecture. Where existing CSMA algorithms are based on ad hoc scheduling rules, virtual time CSMA is a "fuzzy" (because of the nonzero propagation times) approximation to the ideal case—an FCFS $M/G/1$ queueing system. And unlike other CSMA algorithms, which have resisted attempts at finding simple analytical models capable of explaining their behavior (e.g., [6]), we were able to find simple closed-form expressions for bounding the delay in virtual time CSMA.

Appendix

Here we wish to solve the model for slotted nonpersistent CSMA with geometric retransmissions discussed in Section VII. The system operates in discrete time, with arrival points occurring once every a time units. Each station can be "thinking" (trying to generate a new message), "transmitting," or "blocked" (waiting to transmit a message); state transitions can occur at each arrival point. Each thinking station generates a new message with probability σ and either begins transmitting it, if the channel is idle, or enters the blocked state otherwise. Each transmitting station remains in that state for one transmission time (i.e., $1/a + 1$ arrival points), then enters the thinking state, if its transmission was successful, or the blocked state otherwise. If the channel is idle, each blocked station begins transmitting its message with probability ν.

Since the time spent in the "transmitting" state is deterministic, the number of stations in each state at each arrival point exhibits memory. However, an embedded Markov chain can be found by examining the state of the system at the start of each *transmission cycle* [25]. The jth transmission cycle consists of the jth channel idle interval followed by the jth channel busy interval. Since no stations are transmitting when the channel is idle, the state of the system at each embedding point is simply the number of messages in the system (or, equivalently, the number of stations in the blocked state), which has $\{\pi_i\}$ as its steady-state distribution. Let

$$\sigma_{(k)} \triangleq 1-(1-\sigma)^k \tag{A.1}$$

be the probability that a thinking station generates a new message within k arrival points. Then

$$f_k(j,\, i)=\binom{M-j}{i}\, \sigma_{(k)}{}^i(1-\sigma_{(k)})^{M-j-i} \quad 0\le i\le M-j \tag{A.2}$$

is the probability that when j stations are blocked, i thinking stations generate new messages within a period of k arrival points, and

$$g(j,\, i)=\binom{j}{i}\, \nu^i(1-\nu)^{j-1} \qquad 0\le i\le j \tag{A.3}$$

is the probability that when the channel is idle and j stations are blocked, i of them transmit in the same arrival point. It is easy to see that the j, kth element of the transition matrix for the embedded Markov chain satisfies

$$\begin{aligned} p_{j,k}=\frac{1}{1-f_1(j,\,0)g(j,\,0)}\Big(&f_1(j,\,1)g(j,\,0)f_{1/a}(j+1,\,k-j)\\ &+f_1(j,\,0)g(j,\,1)f_{1/a}(j,\,k-j+1)\\ &+f_1(j,\,0)[1-g(j,\,0)-g(j,\,1)]f_{1/a}(j,\,k-j)\\ &+f_1(j,\,1)[1-g(j,\,0)]f_{1/a}(j+1,\,k-j-1)\\ &+\sum_{i=2}^{k-j} f_1(j,\,i)f_{1/a}(j+i,\,k-j-i)\Big) \end{aligned} \tag{A.4}$$

for $k = j-1, j, \cdots, M$. It follows that we can find $\{\pi_j\}$ up to a multiplicative constant using the recurrence relation

$$\pi_j=\frac{\pi_{j-1}(1-p_{j-1,j-1})-\sum_{i=0}^{j-2}\pi_i p_{i,j-1}}{p_{j,j-1}} \qquad j=1,\,2,\,\cdots \tag{A.5}$$

and an initial estimate for π_0. (For numerical stability, it is best to adjust this constant dynamically to reduce the risk of underflows and overflows.)

Having found $\{\pi_j\}$, we can express the average number of messages in system, $\bar{N}$, as

$$\bar{N}=\frac{\sum_{j=0}^{M}\pi_j\bar{N}_jL_c{}^{(j)}}{\sum_{j=0}^{M}\pi_jL_c{}^{(j)}} \tag{A.6}$$

where $\bar{N}_j$ is averaged over all transmission cycles that began with j blocked stations, and $L_c{}^{(j)}$ is the average duration of such transmission cycles. It can be shown [25] that

$$L_c{}^{(j)}=\frac{a}{1-f_1(j,\,0)g(j,\,0)}+1+a. \tag{A.7}$$

To find $\bar{N}_j$, we must account for the messages that are generated part way through a channel busy interval. Let $\Psi_i(t)$ be the average number of messages in the system during t arrival points, given that there are i such messages initially and no transitions to the thinking state take place. Then

$$\begin{aligned}\Psi_i(t)&=i+\frac{M-i}{t}\sum_{j=0}^{t}(t-j)(1-\sigma)^j\sigma\\ &=M-\frac{(M-i)(1-(1-\sigma)^t)(1-\sigma)}{t\sigma}.\end{aligned} \tag{A.8}$$

Thus, if k messages were in the system at the start of a channel busy interval, the average number of messages in the system over the remainder of that channel busy interval is given by $\Psi_k(1/a)$. But

$$\hat{p}_{j,k}=\begin{cases}\dfrac{[1-g(j,\,0)]f_1(j,\,0)}{1-g(j,\,0)f_1(j,\,0)} & k=j\\[2ex] \dfrac{f_1(j,\,k-j)}{1-g(j,\,0)f_1(j,\,0)} & k>j\end{cases} \tag{A.9}$$

is the probability of there being k messages in the system at the start of a channel busy interval, given that we were in state j at the start of the transmission cycle. Thus,

$$\begin{aligned}\bar{N}_jL_c{}^{(j)}&=\frac{ja}{1-f_1(j,\,0)g(j,\,0)}+\sum_{k=j}^{M}\hat{p}_{j,\,k}[ka+\Psi_k(1/a)]\\ &=\frac{a[j+(1+Z)(M-j)\sigma]}{1-f_1(j,\,0)g(j,\,0)}+ai+M-aMZ\end{aligned} \tag{A.10}$$

where $Z = (1-\sigma)(1-(1-\sigma)^{1/a})/\sigma$.

To find the steady-state throughput S, it remains to find $H_c{}^{(j)}$, the expected number of successful transmissions per transmission cycle, given that we were in state j at the start of the transmission cycle. But this is clearly

$$H_c{}^{(j)}=\frac{f_1(j,\,1)g(j,\,0)+f_1(j,\,0)g(j,\,1)}{1-f_1(j,\,0)g(j,\,0)} \tag{A.11}$$

so that the throughput may be expressed as

$$S=\frac{\sum_{j=0}^{M}\pi_jH_c{}^{(j)}}{\sum_{j=0}^{M}\pi_jL_c{}^{(j)}} \tag{A.12}$$

and, using Little's result, the mean delay T is given by

$$T=\frac{\sum_{j=0}^{M}\pi_j\bar{N}_jL_c{}^{(j)}}{\sum_{j=0}^{M}\pi_jH_c{}^{(j)}}. \tag{A.13}$$

References

[1] N. Abramson, "The ALOHA system—Another alternative for computer communications," in *AFIPS Conf. Proc., Fall Joint Comput. Conf.*, 1970, vol. 37, pp. 281-285.

[2] J. I. Capetanakis, "Generalized TDMA: The multi-accessing tree protocol," *IEEE Trans. Commun.*, vol. COM-27, pp. 1476–1484, Oct. 1979.

[3] G. Fayolle, E. Gelenbe, and J. Labetoulle, "Stability and optimal control of the packet switching broadcast channel," *J. ACM*, vol. 24, pp. 375–386, July 1977.

[4] M. J. Ferguson, "On the control, stability and waiting time in a slotted ALOHA random access system," *IEEE Trans. Commun.*, vol. COM-23, Nov. 1975.

[5] R. G. Gallager, "Conflict resolution in random access broadcast networks," in *Proc. AFOSR Workshop Commun. Theory Appl.*, Sept. 17–20, 1978, pp. 74–76.

[6] E. Gelenbe and I. Mitrani, "Control policies in CSMA local area networks: Ethernet controls," in *Proc. ACM SIGMETRICS Conf. Meas. Modeling Comput. Syst.*, Aug. 1982, pp. 223–240.

[7] J. F. Hayes, "An adaptive technique for local distribution," *IEEE Trans. Commun.*, vol. COM-26, Aug. 1978.

[8] R. E. Kahn, S. A. Gronemeyer, J. Burchfiel, and R. C. Kunzelman, "Advances in packet radio technology," *Proc. IEEE*, vol. 66, pp. 1468–1496, Nov. 1978.

[9] L. Kleinrock, *Queueing Systems, Vol. I, Theory.* New York: Wiley-Interscience, 1975.

[10] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels: Part I—Carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Trans. Commun.*, vol. COM-23, pp. 1400–1416, Dec. 1975.

[11] L. Kleinrock, *Queueing Systems, Vol. II, Computer Applications.* New York: Wiley-Interscience, 1976.

[12] D. Konstantas, "Virtual time CSMA: A study," M.Sc. thesis, Dep. Comput. Sci., Comput. Syst. Res. Group, Univ. Toronto, Toronto, Ont., Canada, Tech. Rep. CSRG-149, Jan. 1983.

[13] S. S. Lam and L. Kleinrock, "Packet switching in a multiaccess broadcast channel: Dynamic control procedures," *IEEE Trans. Commun.*, vol. COM-23, pp. 891–904, Sept. 1975.

[14] J. Little, "A proof of the queueing formula $L = \lambda W$," *Oper. Res.*, vol. 9, pp. 383–387, Mar. 1961.

[15] J. L. Massey, "Collision-resolution algorithms and random-access communications," School Eng. Appl. Sci., Univ. California, Los Angeles, Rep. UCLA-ENG-8016, Apr. 1980.

[16] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Commun. ACM*, vol. 19, July 1976.

[17] M. L. Molle, "Unifications and extensions of the multiple access communications problem," Ph.D. dissertation, Dep. Comput. Sci., Univ. California, Los Angeles, CSD Rep. 810730 (UCLA-ENG-8118), July 1981.

[18] ——, "Asynchronous multiple access tree algorithms," in *Proc. ACM SIGCOMM '83 Symp. Commun. Architectures, Protocols,* March 1983; preprint: Comput. Syst. Res. Group, Univ. Toronto, Toronto, Ont., Canada, Tech. Rep. CSRG-145, Aug. 1982.

[19] M. L. Molle and D. Konstantas, "A simulation study of retransmission strategies for the asynchronous virtual-time CSMA protocol," in *Proc. Performance '83, 9th Int. Symp. Comput. Perform. Modeling, Meas., Eval.*, May 1983, pp. 295–308.

[20] G. C. Polyzos, "Tree conflict resolution algorithms: The nonhomogeneous case,"M.A.Sc. thesis, Dep. Elec. Eng., Univ. Toronto, Toronto, Ont., Canada, Dec. 1984.

[21] E. G. Rawson and R. M. Metcalfe, "Fibernet: Multimode optical fibers for local computer networks," *IEEE Trans. Commun.*, vol. COM-26, pp. 983–990, July 1978.

[22] L. G. Roberts, "ALOHA packet system with and without slots and capture," ARPA Network Inform. Cen., Stanford Res. Inst., Menlo Park, CA, ASS Note 8 (NIC 11290), June 1972; reprinted in *Comput. Commun. Rev.*, vol. 5, pp. 28–42, Apr. 1975.

[23] K. C. Sevcik, private communications.

[24] A. S. Tanenbaum, *Computer Networks.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

[25] F. A. Tobagi, "Random access techniques for data transmission over packet switched radio networks," Ph.D. dissertation, Dep. Comput. Sci., Univ. California, Los Angeles, UCLA-ENG-7499, Dec. 1974.

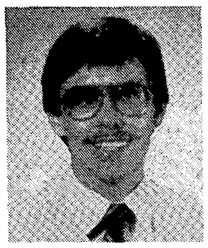
[26] ——, "Multiaccess protocols in packet communication systems," *IEEE Trans. Commun.*, vol. COM-28, pp. 468–488, Apr. 1980.

[27] B. S. Tsybakov and V. A. Mikhailov, "Free synchronous packet access in a broadcast channel with feedback," *Probl. Inform. Transmiss.*, 1980; transl. from Russian, *Probl. Peredachi Inform.*, vol. 14, pp. 32–59, Oct.–Dec. 1978.

[28] ——, "Ergodicity of a slotted Aloha system," *Probl. Inform. Transmiss.*, 1980; transl. from Russian.

[29] ——, "Random multiple packet access: Part-and-try algorithm," *Probl. Inform. Transmiss.*, 1981; transl. from Russian, *Probl. Peredachi Inform.*, vol. 16, pp. 65–79, Oct.–Dec. 1980.

★



Mart L. Molle (S'80–M'82) was born in Toronto, Ont., Canada, on November 2, 1953. He received the B.Sc.(Hons.) degree in mathematics/computing science from Queen's University, Kingston, Ont., in 1976, and the M.S. and Ph.D. degrees in computer science from the University of California at Los Angeles in 1978 and 1981, respectively.

He is now on the faculty of the Department of Computer Science at the University of Toronto, Toronto, where he also holds a cross-appointment in the Department of Electrical Engineering and is a member of the Computer Systems Research Institute. His research interests include the modeling and analysis of protocols for computer networks and distributed systems.

Dr. Molle is a member of the Association for Computing Machinery.

★

Leonard Kleinrock (S'55–M'64–SM'71–F'73), for a photograph and biography, see p. 638 of the July 1985 issue of this TRANSACTIONS.